

# **International Journal on Advances in Security**



The *International Journal on Advances in Security* is published by IARIA.

ISSN: 1942-2636

journals site: <http://www.iariajournals.org>

contact: [petre@iaria.org](mailto:petre@iaria.org)

Responsibility for the contents rests upon the authors and not upon IARIA, nor on IARIA volunteers, staff, or contractors.

IARIA is the owner of the publication and of editorial aspects. IARIA reserves the right to update the content for quality improvements.

Abstracting is permitted with credit to the source. Libraries are permitted to photocopy or print, providing the reference is mentioned and that the resulting material is made available at no cost.

Reference should mention:

*International Journal on Advances in Security, issn 1942-2636*  
vol. 5, no. 1 & 2, year 2012, <http://www.iariajournals.org/security/>

The copyright for each included paper belongs to the authors. Republishing of same material, by authors or persons or organizations, is not allowed. Reprint rights can be granted by IARIA or by the authors, and must include proper reference.

Reference to an article in the journal is as follows:

<Author list>, "<Article title>"  
*International Journal on Advances in Security, issn 1942-2636*  
vol. 5, no. 1 & 2, year 2012, <start page>:<end page> , <http://www.iariajournals.org/security/>

IARIA journals are made available for free, proving the appropriate references are made when their content is used.

Sponsored by IARIA

[www.iaria.org](http://www.iaria.org)

Copyright © 2012 IARIA

**Editor-in-Chief**

Reijo Savola, VTT Technical Research Centre of Finland, Finland

**Editorial Advisory Board**

Vladimir Stantchev, Berlin Institute of Technology, Germany  
Masahito Hayashi, Tohoku University, Japan  
Clement Leung, Victoria University - Melbourne, Australia  
Michiaki Tatsubori, IBM Research - Tokyo Research Laboratory, Japan  
Dan Harkins, Aruba Networks, USA

**Editorial Board**

Gerardo Adesso, University of Nottingham, UK  
Ali Ahmed, Monash University, Sunway Campus, Malaysia  
Manos Antonakakis, Georgia Institute of Technology / Damballa Inc., USA  
Afonso Araujo Neto, Universidade Federal do Rio Grande do Sul, Brazil  
Reza Azarderakhsh, The University of Waterloo, Canada  
Ilija Basicovic, University of Novi Sad, Serbia  
Francisco J. Bellido Outeiriño, University of Cordoba, Spain  
Farid E. Ben Amor, University of Southern California / Warner Bros., USA  
Jorge Bernal Bernabe, University of Murcia, Spain  
Lasse Berntzen, Vestfold University College - Tønsberg, Norway  
Jun Bi, Tsinghua University, China  
Catalin V. Birjoveanu, "Al.I.Cuza" University of Iasi, Romania  
Wolfgang Boehmer, Technische Universitaet Darmstadt, Germany  
Alexis Bonnetaze, Université d'Aix-Marseille, France  
Carlos T. Calafate, Universitat Politècnica de València, Spain  
Juan-Vicente Capella-Hernández, Universitat Politècnica de València, Spain  
Zhixiong Chen, Mercy College, USA  
Peter Cruickshank, Edinburgh Napier University Edinburgh, UK  
Nora Cuppens, Institut Telecom / Telecom Bretagne, France  
Glenn S. Dardick, Longwood University, USA  
Vincenzo De Florio, University of Antwerp & IBBT, Belgium  
Paul De Hert, Vrije Universiteit Brussels (LSTS) - Tilburg University (TILT), Belgium  
Pierre de Leusse, AGH-UST, Poland  
Raimund K. Ege, Northern Illinois University, USA  
Laila El Aïmani, Technicolor, Security & Content Protection Labs., Germany  
El-Sayed M. El-Alfy, King Fahd University of Petroleum and Minerals, Saudi Arabia  
Rainer Falk, Siemens AG - Corporate Technology, Germany  
Shao-Ming Fei, Capital Normal University, Beijing, China

Eduardo B. Fernandez, Florida Atlantic University, USA  
Anders Fongen, Norwegian Defense Research Establishment, Norway  
Somchart Fugkeaw, Thai Digital ID Co., Ltd., Thailand  
Steven Furnell, University of Plymouth, UK  
Clemente Galdi, Universita' di Napoli "Federico II", Italy  
Marco Genovese, Italian Metrological Institute (INRIM) -Torino, Italy  
Birgit F. S. Gersbeck-Schierholz, Leibniz Universität Hannover, Certification Authority University of Hannover (UH-CA), Germany  
Manuel Gil Pérez, University of Murcia, Spain  
Karl M. Goeschka, Vienna University of Technology, Austria  
Stefanos Gritzalis, University of the Aegean, Greece  
Michael Grottke, University of Erlangen-Nuremberg, Germany  
Ehud Gudes, Ben-Gurion University - Beer-Sheva, Israel  
Indira R. Guzman, Trident University International, USA  
Huong Ha, University of Newcastle, Singapore  
Petr Hanáček, Brno University of Technology, Czech Republic  
Gerhard Hancke, Royal Holloway / University of London, UK  
Sami Harari, Institut des Sciences de l'Ingénieur de Toulon et du Var / Université du Sud Toulon Var, France  
Dan Harkins, Aruba Networks, Inc., USA  
Ragib Hasan, University of Alabama at Birmingham, USA  
Masahito Hayashi, Nagoya University, Japan  
Michael Hobbs, Deakin University, Australia  
Neminath Hubballi, Infosys Labs Bangalore, India  
Mariusz Jakubowski, Microsoft Research, USA  
Ángel Jesús Varela Vaca, University of Seville, Spain  
Ravi Jhavar, Università degli Studi di Milano, Italy  
Dan Jiang, Philips Research Asia Shanghai, China  
Georgios Kambourakis, University of the Aegean, Greece  
Florian Kammüller, Middlesex University - London, UK  
Sokratis K. Katsikas, University of Piraeus, Greece  
Seah Boon Keong, MIMOS Berhad, Malaysia  
Sylvia Kierkegaard, IAITL-International Association of IT Lawyers, Denmark  
Marc-Olivier Killijian, LAAS-CNRS, France  
Hyunsung Kim, Kyungil University, Korea  
Ah-Lian Kor, Leeds Metropolitan University, UK  
Evangelos Kranakis, Carleton University - Ottawa, Canada  
Lam-for Kwok, City University of Hong Kong, Hong Kong  
Jean-Francois Lalande, ENSI de Bourges, France  
Gyungho Lee, Korea University, South Korea  
Clement Leung, Hong Kong Baptist University, Kowloon, Hong Kong  
Diego Liberati, Italian National Research Council, Italy  
Giovanni Livraga, Università degli Studi di Milano, Italy  
Gui Lu Long, Tsinghua University, China  
Jia-Ning Luo, Ming Chuan University, Taiwan  
Thomas Margoni, University of Western Ontario, Canada  
Rivalino Matias Jr ., Federal University of Uberlandia, Brazil

Manuel Mazzara, UNU-IIST, Macau / Newcastle University, UK  
Carla Merkle Westphall, Federal University of Santa Catarina (UFSC), Brazil  
Ajaz H. Mir, National Institute of Technology, Srinagar, India  
Jose Manuel Moya, Technical University of Madrid, Spain  
Leonardo Mostarda, Middlesex University, UK  
Jogesh K. Muppala, The Hong Kong University of Science and Technology, Hong Kong  
Syed Naqvi, CETIC (Centre d'Excellence en Technologies de l'Information et de la Communication), Belgium  
Sarmistha Neogy, Jadavpur University, India  
Mats Neovius, Åbo Akademi University, Finland  
Jason R.C. Nurse, University of Oxford, UK  
Peter Parycek, Donau-Universität Krems, Austria  
Konstantinos Patsakis, Rovira i Virgili University, Spain  
João Paulo Barraca, University of Aveiro, Portugal  
Sergio Pozo Hidalgo, University of Seville, Spain  
Vladimir Privman, Clarkson University, USA  
Yong Man Ro, KAIST (Korea advanced Institute of Science and Technology), Korea  
Rodrigo Roman Castro, Institute for Infocomm Research (Member of A\*STAR), Singapore  
Heiko Roßnagel, Fraunhofer Institute for Industrial Engineering IAO, Germany  
Claus-Peter Rückemann, Leibniz Universität Hannover / Westfälische Wilhelms-Universität Münster / North-German Supercomputing Alliance, Germany  
Antonio Ruiz Martinez, University of Murcia, Spain  
Paul Sant, University of Bedfordshire, UK  
Reijo Savola, VTT Technical Research Centre of Finland, Finland  
Peter Schartner, University of Klagenfurt, Austria  
Alireza Shameli Sendi, Ecole Polytechnique de Montreal, Canada  
Dimitrios Serpanos, Univ. of Patras and ISI/RC ATHENA, Greece  
Pedro Sousa, University of Minho, Portugal  
George Spanoudakis, City University London, UK  
Lars Strand, Nofas, Norway  
Young-Joo Suh, Pohang University of Science and Technology (POSTECH), Korea  
Jani Suomalainen, VTT Technical Research Centre of Finland, Finland  
Enrico Thomae, Ruhr-University Bochum, Germany  
Tony Thomas, Indian Institute of Information Technology and Management - Kerala, India  
Panagiotis Trimintzios, ENISA, EU  
Peter Tröger, Hasso Plattner Institute, University of Potsdam, Germany  
Simon Tsang, Applied Communication Sciences, USA  
Marco Vallini, Politecnico di Torino, Italy  
Bruno Vavala, Carnegie Mellon University, USA  
Mthulisi Velempini, North-West University, South Africa  
Miroslav Velev, Aries Design Automation, USA  
Salvador E. Venegas-Andraca, Tecnológico de Monterrey / Texia, SA de CV, Mexico  
Szu-Chi Wang, National Cheng Kung University, Tainan City, Taiwan R.O.C.  
Piyi Yang, University of Shanghai for Science and Technology, P. R. China  
Rong Yang, Western Kentucky University, USA  
Hee Yong Youn, Sungkyunkwan University, Korea

Bruno Bogaz Zarpelao, University of Campinas (UNICAMP), Brazil  
Wenbing Zhao, Cleveland State University, USA

## CONTENTS

*pages 1 - 15*

### **Application of Scenario-driven Role Engineering in Knowledge Management Systems - Requirements and Implementation**

Daniel Kimmig, Karlsruhe Institute of Technology, Germany  
Andreas Schmidt, University of Applied Sciences, Karlsruhe, Germany  
Klaus Bittner, Karlsruhe Institute of Technology, Germany  
Markus Dickerhof, Karlsruhe Institute of Technology, Germany

*pages 16 - 27*

### **A Scalability Analysis of an Architecture for Countering Network-Centric Insider Threats**

Faisal Sibai, George Mason University, USA  
Daniel Menasce, George Mason University, USA

*pages 28 - 35*

### **Advances in Protecting Remote Component Authentication**

Rainer Falk, Siemens AG Corporate Technology, Germany  
Steffen Fries, Siemens AG Corporate Technology, Germany

*pages 36 - 45*

### **A Privacy Preserving Solution for Webmail Systems with Searchable Encryption**

Karthick Ramachandran, University of Western Ontario, Canada  
Hanan Lutfiyya, University of Western Ontario, Canada  
Mark Perry, University of Western Ontario, Canada

*pages 46 - 67*

### **Organizing Security Patterns Related to Security and Pattern Recognition Requirements**

Michaela Bunke, Center for Computing Technologies (TZI), Universität Bremen, Germany  
Rainer Koschke, Center for Computing Technologies (TZI), Universität Bremen, Germany  
Karsten Sohr, Center for Computing Technologies (TZI), Universität Bremen, Germany

# Application of Scenario-driven Role Engineering in Knowledge Management Systems - Requirements and Implementation

Daniel Kimmig\*, Andreas Schmidt<sup>†</sup>\*, Klaus Bittner\*, and Markus Dickerhof\*

\*Institute for Applied Computer Science

Karlsruhe Institute of Technology

Karlsruhe, Germany

E-mail: {daniel.kimmig, andreas.schmidt, klaus.bittner, markus.dickerhof}@kit.edu

<sup>†</sup>Department of Informatics and Business Information Systems,

University of Applied Sciences, Karlsruhe

Karlsruhe, Germany

E-mail: andreas.schmidt@hs-karlsruhe.de

**Abstract**—Collaborative systems, which are often used in short-term virtual enterprises or long-term cooperation networks, often contain information about the manufacturing and fabrication competences of the participating technology partners. These should only be made available to a very restricted group of persons for example to support feasibility studies in the context of actual customer requests. This is a new important feature to be supported in nowadays knowledge management systems. Hence, the goal of this paper is to present a methodology for implementing an access control mechanism based on role based access control. This mechanism supports the definition of fine granular access rights capable of protecting sensible information often found in cooperative process knowledge management systems. In this paper we will discuss models of access control and present an adaption of the scenario-driven role engineering method to the special needs in a collaborative process knowledge management system with very particular access requirements. Beside the adaption of the scenario-driven role engineering method to such a system, the adapted method will be concretely applied to the process knowledge management system *MinaBASE*, which was developed in our institute. To complete, an implementation will be shown with the help of the inversion of control framework “Spring Security” as well as aspect-oriented programming. Here static as well as dynamic aspects of security will be presented. The paper shows in a detailed manner the usability of the scenario-driven role engineering method for applications in the field of collaborative knowledge management.

**Keywords**—Access control; Knowledge Management; RBAC; Role Engineering.

## I. INTRODUCTION

Both corporate groups as well as small and medium-sized enterprises (SME) are experiencing increasing competition and shorter product lifecycles [1]. The resulting necessity of shortening the product development process also has to be mastered by enterprises in the field of microsystems technologies (MST) that are characterized by a high interdisciplinarity and complex, multi-stage, and hardly standardized fabrication processes. Frequently, every product is produced by an individually tailored fabrication process [2].

While larger MST enterprises still manage a wide spectrum of technologies, SME rather tend to offer solutions in a high specialized area. To offer more complex solutions, they establish technical partnerships with other SME. These may have the form of short-term virtual enterprises or long-term cooperation networks. To support such organization forms in the field of MST, the *MinaBASE* process knowledge database was developed by the Institute for Applied Computer Science of Karlsruhe Institute of Technology. By means of this database, the manufacturing and fabrication competences of the technology partners are made available to a central coordinator who then assesses technical and economic feasibility.

This information, however, includes company secrets, whose confidentiality and integrity is of crucial importance to a company’s existence. Acceptance of *MinaBASE* therefore does not only depend on meeting functional requirements, but also on aspects like security and access protection. To prevent an undesired disclosure of company secrets, access shall be controlled by the established role-based access control (RBAC) [3]. Here, authorizations are not assigned directly to subjects, i.e., the users of a system, but to abstract roles to which the users are assigned subsequently. In this way, the frequently error-prone maintenance is reduced and security is increased. This requires the definition of an adequate role concept.

Information systems often use standard models with system-wide administrators, owners of information objects, and guests having restricted read access. While this is a reasonable default for establishing a minimal level of access control, it does not consider the business processes in which the system is used and which particular requirements result in terms of confidentiality and data integrity. The lack of a role concept tailored to these specific requirements can severely harm the acceptance and usage of a knowledge management system. For example in *MinaBASE*, it should be possible to grant a partner access to product-related



properties of a microsystem during the sales process without disclosing the configuration of machine parameters to produce these properties. Such an application exceeds the modeling capabilities of the standard role concept described above, as external partners are not the owners of this information. This shows that nowadays knowledge management systems have particular security requirements, which require activities of role engineering to create an adequate role concept to support usage in different business processes. Our contribution takes on this problem by showing how a role concept tailored to the particular requirements of process knowledge management systems can be defined and implemented. The main contribution of the paper is twofold: First, the scenario-driven role engineering method will be adapted to the requirements of collaborative knowledge management systems. And second, the suitability of existing access frameworks to implement the adapted method will be shown by means of the framework spring security.

The paper is structured as follows: In the next section, the *MinaBASE* process knowledge database will be presented. Then, mechanisms for access control as well as the scenario-driven role engineering approach [4] and the adaptations due to the background and objective of *MinaBASE* will be outlined. Subsequently, the methodology will be applied and a role concept for RBAC ensuring data integrity and confidentiality for *MinaBASE* will be derived. The final section describes the implementation of this role concept within an "Inversion-of-Control"-Framework (IoC) by demonstrating how Spring Security and technologies such as aspect-oriented programming (AOP) can be used to fulfill static and dynamic security requirements.

## II. *MinaBASE* PROCESS KNOWLEDGE DATABASE

The knowledge required to produce added value is no public property, but a company resource that has to be administrated efficiently in order to ensure economic success. To support this process, knowledge management systems have been established [5]. In process-oriented knowledge management [6], these methods are applied to highly knowledge-intensive fabrication processes, as those used in MST. The *MinaBASE* process knowledge database is used by the technology partners for the structured storage of technical fabrication parameters of the methods and materials used in MST and of the partner-specific technical competences. In *MinaBASE*, the smallest information entity is the so-called technical aspect (TA). It is used to model materials, machines, and fabrication technologies [7]. By means of generalization hierarchies, TAs are arranged in taxonomies. The number and contents of taxonomy trees can be specified and modified during runtime, such that a flexible structure meeting MST requirements can be defined for the storage of fabrication know-how. TAs may be assigned properties referred to as technical parameters (TP). A TP is a string of characters, integers, or floating-point numbers in a certain

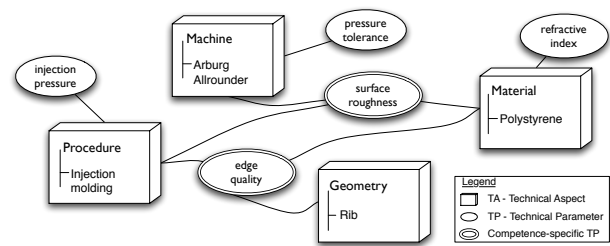


Figure 1. Schematic representation of a *MinaBASE* competence [9].

unit and references an attribute, e.g., the aspect ratio. The TP of a TA are inherited by lower partial hierarchies of the hierarchy tree in analogy to the object-oriented approach. In addition, lower hierarchy levels can further refine the inherited TP. Classification of TP places them in a certain context, such that a TP refers to a product or its fabrication and, hence, is either product-specific or fabrication-specific. Product-specific TP describe the properties of a microsystem, such as the depth of a groove reached by the fabrication process of milling. Fabrication-specific TP refer to the machine configuration needed to produce a specific product property. For modeling the capabilities of a technology partner, competences [8] are considered to be a set of various TA from disjunct hierarchy trees, which is illustrated in Figure 1. This figure schematically represents the competence "injection molding of a rib with polystyrene using the Arburg Allrounder machine" together with some TP. From the hierarchy trees of process, machine, material, and geometry element, the TAs are selected. These TAs are characterized by their TP, such as the injection pressure of the injection molding process. The combination of these TAs results in the competence that is reflected by other TP, such as the edge quality and surface roughness. Consequently, a competence is a type of view of a certain combination of TAs with properties in the form of TP that apply to this combination only, i.e., that characterize the competence in more detail. TAs can be used in several competences. They represent reusable, encapsulated, smallest information entities. An extension of the *MinaBASE* concept has been developed in order to reuse these information entities to allow process modeling of manufacturing sequences based on semantic technologies [9].

## III. ACCESS CONTROL

Information security is concerned with mitigating risks that affect information systems in the age of growing interconnectedness of computers. The purpose of information security is to establish a state, in which the following three criteria are met for the protected information.

- *Confidentiality* in this context means that only users with certain privileges are allowed to access protected information.

- *Integrity* ensures that information can only be altered or deleted by users that have sufficient permissions.
- *Availability* is the requirement that an authorized access to information is possible at any time.

Access control mainly refers to the first two points above: Confidentiality and integrity. But it also has an impact on the third criterion, since bypassing access control is often a first step to be able to compromise availability [10]. In the digital age massive productivity gains have been achieved due to the integrated availability of enterprise data in business information systems. However, the availability of critical data in such systems involves risks, because the majority of attacks come from within an organization itself [11]. This means that in the long term value added can only be achieved when access to the information systems can be controlled appropriately. The so-called "Broken Access Control", i.e., the selective exploitation of unsuitable access control, takes a central place in the OWASP (Open Web Application Security Project - a project, which focuses on the analysis and control of secure software), which reports the most common vulnerabilities of web based applications. From this it can be concluded that the design of an optimal access control for applications such as *MinaBASE* is one of the most important factors for ensuring information security. Essential for this are principles for the design of secure applications that are described briefly below.

- The principle *Fail-safe defaults* stipulates that any attempt to access any object by an arbitrary subject is to be rejected, unless it was explicitly permitted [12] [13]. Instead of asking why one cannot access a resource, it is more important to ask why one should be able to access it in the first place.
- The *Principle of Least Privilege* was first postulated by Saltzer and Schroeder [14] and states that a user within a certain period of time, e.g., during a session, should only possess the minimal amount of privileges that are sufficient for him to fulfill his task. This will ensure that access control cannot be easily circumvented by privileges that were granted too loosely [10].
- The demand for the principle of *Separation of Duty* (SoD) is correlated to the need for integrity of information. This mainly concerns operations on resources that are either very risky or where a situation cannot be excluded in which resources are at risk of being abused even by authorized users. For such cases SoD suggests to split the operation onto multiple users, so that no single user has sufficient authority to that operation [10].

After the objectives and key principles for information security through access control have been introduced, the following sections will present different models of designing access control mechanisms, which are qualified to achieve the described goals of information security.

#### A. Discretionary Access Control

The model of *Discretionary Access Control* (DAC) is based on the principle of "object ownership", which means that one or more owners are assigned a resource, grant permission to access these resources at their discretion. The decision on granting access is based on the identity of a user or his group membership [10]. The term "discretionary" means that the passing of permissions to specific operations on resources such as files is determined by the owner's confirmation. This means that each user is enabled to spread these rights to another user or his subjects [10]. The implementation of the permission distribution is based on so-called "Access Control Lists" (ACLs) or capability lists. The advantage of DAC is a high degree of flexibility, since it is possible to grant permissions in a very fine-grained manner. The drawback of DAC is that you can not limit the spread of permissions.

#### B. Mandatory Access Control

*Mandatory Access Control* (MAC) relies on weaknesses of DAC, and is therefore specifically designed for the containment of potential information flows [15]. This is achieved as MAC has no principle of object-ownership, but is built upon a classification of information. The sensitivity of information and user status function as a decision criterion for access requests. The sensitivity is determined by the classification of the information depending on how big the damage caused by the loss of confidentiality would be. MAC prevents the proliferation of permissions to users who were not considered to be authorized, as the mechanisms of MAC protect information of a certain level from being accessed by users of an insufficient level. Examples of such mechanisms are the *Bell-LaPadula model* [16] and the *Biba Integrity Model* [17]. The advantage of MAC is the safe limitation of access permissions, however it prevents the flexible sharing of information between users, because the classification of information and users is predetermined and therefore static and rather inflexible [13].

#### C. Role-based Access Control

In *Role-Based Access Control*, permissions for operations on resources are not assigned directly to users, but an abstraction in between both concepts is created, which is referred to as a role [3]. The meaning of these roles is directly comparable with roles in organizations. Subjects who use a system to accomplish similar tasks, act in a similar role towards the system. Therefore completely different permissions are required, which are limited by the operations necessary to accomplish the different tasks. Permissions are assigned to roles in RBAC [10], thus there is no direct allocation of permissions between users and objects. The reason for the development of the RBAC model is based on two insights: The first is that after investigating the security requirements of commercial organizations, it was

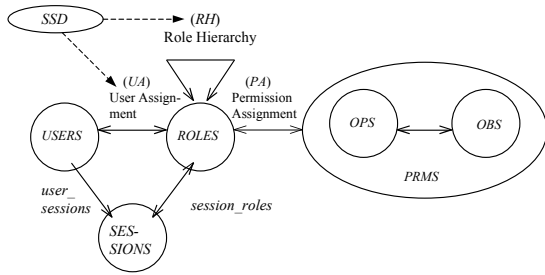


Figure 2. Schematic overview of the extended RBAC model.

found that neither MAC nor DAC cover the needs of these organizations. The origin of the MAC model is the protection of classified information, which means a way of controlling which subject can see what kind of information. In contrast to that, RBAC is concerned with the question of which subjects are allowed to perform which operations on what kind of resources [3]. In addition, it is very difficult to classify information and subjects for a commercial organization, since such a classification is static and inflexible. This lack of flexibility is overcome by the characteristics of DAC to make the decision to limit propagation of access permissions at the discretion of the "Object Owner", since a more dynamic access control becomes possible. The challenge in commercial organizations is, users are not the owners of resources, but the institution, in which they are embedded into [18].

Consequently, the essential principle of "Object Ownership" of DAC is not applicable, as the distribution of access privileges should not be put at the discretion of the users. For this reason RBAC is also referred to as *non-discretionary* [3]. RBAC is less focused on the grouping of users such as DAC, but rather on grouping of permission sets, which enable the execution of operations on resources [3]. Through this concept, the grouping of permissions to roles, administration becomes easier, as changes to users only result in updating the membership to associated roles [18].

It also supports the distribution of RBAC permissions according to the "Principle of Least Privilege", since the roles of an organization can be assigned with a minimal amount of privileges necessary to complete the respective tasks within the organization. If there are conflicts of interest between certain units of the organization, these can be overcome using the technique of "Separation of Duty", which means that restrictions are placed on the distribution of roles. Over time, different stages of RBAC have been developed, which build on each other and will be briefly described below. A schematic structure of these models is given by Figure 2 from [11]. Basic RBAC only consists of three sets, which model users, roles and permissions. Roles exist purely because of the grouped assignment of permissions. In reality, however, there is a hierarchical arrangement, as

some roles consist of more permissions than others and thus the privileges of these are included redundantly. The introduction of a hierarchical arrangement for *Hierarchical RBAC* directly in the model decreases the required effort to administrate access control. The arrangement of roles can then be represented by a partial order as a graph or as an inverted tree. With the idea to encapsulate the functions of an organization and the necessary permissions to roles, it becomes apparent that this can easily lead to an abuse of privileges in a commercial environment. To avoid such cases, the principle of "Separation of Duty" is part of the *Constrained RBAC* model, in order to assign permissions to different roles in cases of conflicts of interest. The use of the RBAC model has the potential to reduce complexity and error rate of access control as well as to reduce the cost and duration of administration.

#### IV. SCENARIO-DRIVEN ROLE ENGINEERING

The term of role engineering (RE) in the context of RBAC means the design and specification of roles, authorizations, secondary conditions, and restrictions as well as of a hierarchic role model [19]. RE is used to create a concrete model for RBAC-based access control. In [4], Scenario-driven role engineering (SDRE) is defined as an approach based on scenarios, such as sequences of actions and events from the user's perspectives. This sequence in a scenario can be subdivided into subscenarios and atomic steps of chronological user interaction. Scenarios are embedded in a task, i.e., a problem or a work area, which links the scenarios of a system with its users. The users mostly apply a system to fulfill a task of their work profile or their job description. This structurization into various levels serves to break down a job description of a user into atomic steps, each of which may be associated with an authorization to access a resource. For various types of users, the minimum amount of authorizations required for the execution of the tasks can be derived. In this way, the principle of least privilege [14] is implemented. For documentation, various models are generated by the SDRE approach, which are interlinked in terms of contents and used for the derivation of the role concept. The *scenario model* describes all scenarios and steps, *task definitions* serve to structure scenarios, the *work profile* summarizes tasks for job descriptions. The *permission catalog* lists the individual permissions or authorizations. It may be complemented by a constraint catalog of special limitations [4]. While the permission catalog is focused on static assignments of authorizations to specific resources, constraints describe dynamic conditions, which are evaluated at runtime. Hereinafter, the SDRE process will be described in general. First, the use scenarios of the system are compiled and their actions and events are documented. Then, subscenarios and steps are defined and the authorizations required for them are included in the permission catalog or special limitations are listed in the

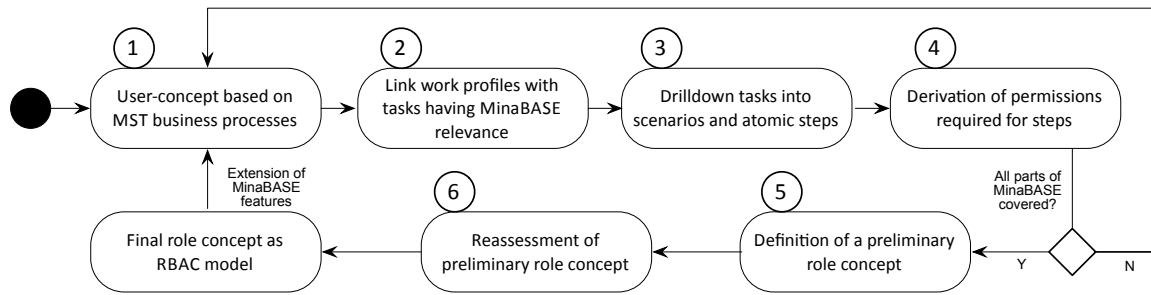


Figure 3. Scenario-driven role engineering according to [4] with adaptations.

constraint catalog. When this step is completed for all scenarios, similar scenarios are generalized. Very complex scenarios are divided into smaller parts which are then included in the scenario model. On this basis, tasks are formed by grouping scenarios. These tasks are then classified into various work profiles. This results in a preliminary role concept and minimum authorizations can be assigned to the individual activities. As a rule, this preliminary model contains duplicates of roles with identical authorizations, which then have to be fused in a last step. This yields the RBAC model as a role concept. The SDRE process represents a systematic approach to RE. It was applied to information systems for the health care sector by the technical committee of HL7 already [20]. Due to this practical test, SDRE in principle may be applied to *MinaBASE*. However, certain adaptations are required, because the background and objective of *MinaBASE* differ from those of the HL7 systems. The paramount objective of *MinaBASE* is the support of knowledge-intensive business processes of MST enterprises by a structurization of the knowledge required for the execution of these business processes. A criterion for the acceptance of knowledge management is its integration in workflows of the users and an efficient and complete coverage of information needs [21]. As such the SDRE approach is to be applied to the use of *MinaBASE* in business processes of MST enterprises and cooperation networks. The model given in [4] is therefore subjected to the following adaptations:

- In the standard SDRE methodology, scenarios for a system are the main input, to which required authorizations are allocated. Subsequently, these scenarios are generalized and assigned to tasks and work-profiles which create a preliminary role concept. For *MinaBASE* however an alternative input is more persuasive. Instead of starting with the scenarios of the system, work areas within the business processes of an MST-company are examined, whether they include tasks in which *MinaBASE* can be used to increase added value. To these tasks scenarios will be assigned in

order to obtain the information, which resources are required for fulfilling them and what authorizations are needed. Based on this information, a preliminary role concept can be derived in a similar way to the standard model due to the minimal set of authorizations for each role. By these adaptations - a switch of input variables to the methodology - the basic principle of SDRE is preserved, while better results for the creation of the role concept are expected, because of the adapted methodology being closer to the business processes of a MST-company.

- The scenarios to be formulated are not based on consequences of actions and events, but will also contain all definable steps. Although these do not occur in sequential order, they can be characterized by a certain access authorization.
- For reasons of clarity, special limitations extending beyond the static allocation of authorizations are included directly in the permission catalog and not in the constraint catalog, such that both models fuse.

The adapted process is illustrated in Figure 3. It comprises six steps, the execution of which shall be described in more detail in the following section.

## V. APPLICATION OF SDRE TO *MinaBASE*

Using parts of the models created by the SDRE process, it shall now be demonstrated how the role concept can be generated systematically.

### A. Step 1: Generation of the User Concept

Application of the model is based on an analysis of the business processes of a model MST company for possibilities of using *MinaBASE* and for activities, where the use of *MinaBASE* can result in a added value. Functions and units of an MST company, which may be potential users of *MinaBASE*, are:

- *Sales, external guest:* *MinaBASE* supports the sales process in the strategic assessment of the feasibility of customer orders, because these decisions can be made

based on an IT documentation of competences and fabrication know-how. Strategic means that a general decision is made without taking into account technical details. In addition, the customer order is typed depending on whether a standard product is to be manufactured or a specification has to be met by enforcing the development in a project. The documented competences can be used as a database for sales promotion. External guests, e.g., customers or suppliers, may be given access to the system in order to stay informed about fabrication processes used by the company or the cooperation network.

- *Project management, development:* If a customer order is classified to be not directly producible by the sales division, a project team is established based on the customer's specification. This team is composed of the project manager and technical experts. In an iterative process, they specify general solution alternatives, the commercial feasibility of which is assessed. In addition, solution approaches, such as functional patterns or prototypes, are developed in detail, the technical feasibility of which is guaranteed. Upon successful agreement with the customer, exact fabrication planning is started in the next step. Planning is based on the results of the development of a commercially and technically feasible solution.
- *Construction, fabrication planning:* Planning of fabrication, i.e., of the individual steps of production flow, may be initiated by a successful development process or a directly producible customer order, e.g., the repetition of an already executed fabrication process. In the latter case, *MinaBASE*, a system for process-oriented knowledge management, provides support by the storage of process elements of process steps and process sections and their combination in process chains, as this allows for the direct use of already executed fabrication processes [9]. This principle in weaker form may also be applied to fabrication planning based on a technical solution alternative from development. By copying or adapting existing process models or process elements, planning of the fabrication process can be accelerated. Construction and fabrication planning result in a detailed schedule for production and defined quality management tests, during which data are measured in the production process.
- *Quality management, production:* Production focuses on the execution of the process steps defined by fabrication planning in a process chain to execute the order placed by the customer. Technicians working at the machines have direct access to production and are capable of using technical parameters of the individual process elements of the process chain for adjusting the machine parameters and of measuring real data during the tests. Various areas of quality management

Work-Profile	Task
Project management (PM)	Compose group of technical experts
	Coordination of orders and manufacturing sequences across boundaries of organizational units regarding process dependencies
	View competences of organizational units
	Analyze process-chains of organizational units
	[...]

Figure 4. Work area project management with associated tasks from the work profile model.

are covered. New fabrication knowledge of attributes and parameters of process elements is generated.

#### B. Step 2: Definition of Work Profiles and Task Definitions

According to the adaptations to the SDRE model, *MinaBASE* tasks are assigned to the enterprise units or work areas listed in the previous section. Figure 4 shows a part of the work profile model. In the work area of project management for the iterative development of solutions for an unsolved development problem of a customer order, tasks are identified, to which the *MinaBASE* resources can be applied. These tasks are the pooling of technical experts, the analysis of fabrication competences and process chains, and the coordination of process dependencies beyond organizational units. The complete work profile model contains all tasks to which *MinaBASE* may be applied. These are the input variables for the detailed assignment of scenarios, steps, and authorizations to access resources in the following step.

#### C. Steps 3/4: Refinement of Scenarios and Assignment of Authorizations

For the first two tasks mentioned in the previous section, namely, the pooling of experts and the coordination of process dependencies, a part of the fused permission and constraint catalog is illustrated in Figure 5. For every scenario or every step, the associated operation on an object or resource is modeled, with R denoting read access (read), C denoting the creation of a new entry (create), U meaning processing (update), and D the deletion (delete) of a resource. The information of which actor accesses which resource with which operation is encapsulated as a permission by a triple of the type (actor, operation, object). At last, limitations or constraints of access are specified. For the first task, the organizational units, contact data of technical experts, and competences of the organizations stored in *MinaBASE* are considered as use scenarios. Read access (R) to the tables of the database and application components is required. The second task is handled similarly, as order data and detailed, production-related attribute values of process dependences are needed. In addition, an entry in the constraint catalog is made to ensure that the actor sees only those attribute

Task	Actor	Scenario/Step	Op	Resource	Permission	Constraint
Compose group of technical experts	PM	View organizational units	R	Enterprise Table	{PL, R, Enterprise Tab}	-
	PM	Inquire contacts (work scheduler)	R	Enterprise-User Table	{PL, R, Enterprise-User Tab}	-
	PM	View competences of organizations	R	Competence-Enterprise Table	{PL, R, Enterprise-Compet Tab}	-
Coordination of orders and manufacturing sequences across boundaries of organizational units regarding process dependencies	PM	View orders	R	Order Table	{PL, R, Order Tab}	-
	PM	View associated organizations of specific orders	R	Order-Enterprise Table	{PL, R, Order-Enterprise Tab}	-
	PM	View associated manufacturing competences of orders	R	Order-Competence Table	{PL, R, Order-Competence Tab}	MR_OE-Filter
	PM	View manufacturing sequence and process dependencies as attribute values	R	Order-Competence-Attribute/Values Table	{PL, R, Order-Competence-Attribute/Values Tab}	MR_OE-Filter

Figure 5. Refinement of tasks in use scenarios and assignment of authorizations to access the resources needed.

values that are characterized as project-specific property and not as production-specific, internal know-how of an organization. This constraint cannot be implemented as a static authorization, as the assignment is made dynamically during runtime.

#### D. Steps 5/6: Derivation of the Role Concept

By applying the first steps of the adapted SDRE model to the identified *MinaBASE*-using enterprise units, a hierarchy corresponding to a preliminary model of the role concept may be derived on the basis of the authorizations. This preliminary model is shown in Figure 6. The highest point is the administration that is not only responsible for administering users and their assignment to roles, but also has all other authorizations in the system. The lowest point is the external guest, who is given fewest access rights. In between, the graph may be structured horizontally and vertically. Vertical structurization results from the degree of orientation to orders. This means that planning of working steps of a process chain and their execution are much more related to orders than the development of solution alternatives for a certain customer specification by technical experts. Horizontal arrangement results from the respective amount of granted privileges.

Then, the last step of the SDRE process follows, i.e., the analysis of the preliminary model for groupings of authorizations in the form of roles that exist several times and have a comparable amount of authorizations. These roles have to be eliminated. Otherwise, the catalog would list more roles than necessary, which might result in anomalies and undesired side effects in the administration of rights and roles. Review of the preliminary model taking into account the criteria described yields the role concept shown in Figure 7. Documentation of the authorizations for the individual company units shows that a separation between project management and development is not reasonable, as the access rights for the modeled scenarios and steps are identical. For this reason, both units are summarized by the developer role. The same applies to fabrication planning and quality management, as both units use *MinaBASE* for various

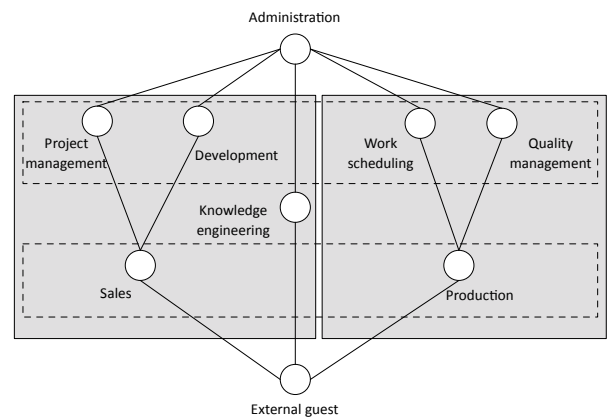


Figure 6. Preliminary role concept based on the permission catalog.

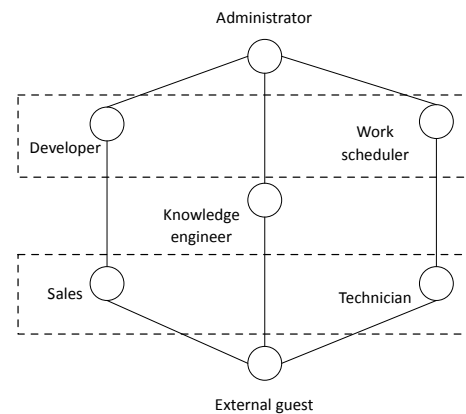


Figure 7. Revised role concept as RBAC model.

objectives, but still have comparable use scenarios and, hence, identical authorizations. Consequently, they assume the same role in the use of *MinaBASE*, the role of the work planner. Underneath the role of the work planner, the role

technician exists, who is responsible for the implementation of the plans made by the work planner. The technician is in the position to acquire the measurement data for the tests of the work planner and to define detailed parameters, such as machine instructions, as information added to process elements. To fulfill his task, the developer needs deep insight into the details of the competences and process chains, as he has to extend the strategic assessment of the sales division by a guaranteed technically possible feasibility assessment. The “knowledge engineering” component has already encapsulated the rights to update order-independent knowledge in the preliminary model. In this way, additional authorizations can be assigned specifically to a role.

## VI. IMPLEMENTATION IN AN IOC-FRAMEWORK

This section describes the implementation of the RBAC model within *MinaBASE*. Firstly, the used framework, Spring Security, is introduced. Subsequently, it is shown how static and dynamic security requirements stemming from the permission catalog as well as the constraint catalog can be fulfilled.

### A. Spring Security

Spring Security is a subproject of the application framework “Spring” to control authentication and authorization in the JEE environment, i.e., in the range of business applications based on Java technology [22]. It is empowered by technologies provided by the core of Spring, such as “Inversion of Control” using “Dependency Injection” (DI), which means a passive provisioning of an application component’s dependencies by a central container known as the Spring “ApplicationContext”. Martin Fowler defined the term dependency injection as means of provisioning an object’s dependencies [23], for which several tools have been developed to aid the construction of large object graphs consisting of many interrelated classes mainly based on declarative configuration. The main motivation behind the idea of dependency injection is the overcoming of drawbacks found in previous solutions to the problem of object graph construction. The “ServiceLocator-Pattern” is an example of such a solution. The difference between the two approaches is displayed in Figure 8.

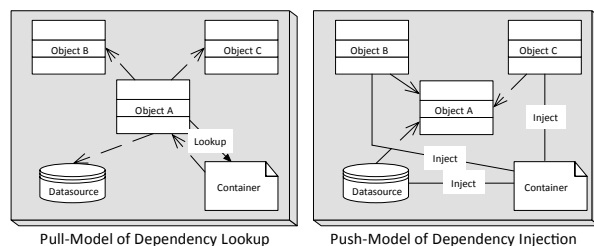


Figure 8. Approaches to constructing object graphs and its dependencies.

The first approach can be characterized as a “Pull-model”, in which an object actively requests certain dependencies from a central container that helps in locating services such as a reference to a datasource. The second approach uses a push-model, where the object is passive and the container will manage the object’s lifecycle alongside the fulfillment of dependencies that the object requires. The main advantage of the second approach is, the object does not need to know about the central container and is therefore easier to test, refactor and maintain [24].

In addition to that, the ApplicationContext provides AOP capabilities. New programming paradigms usually are invented to overcome the weakness of well established paradigms. In the same way AOP is an extension of traditional object-oriented programming (OOP). In OOP, classes are used as blueprints to model objects from the real world. By following principles such as separation of concerns [25] and information hiding of implementation details, OOP had a drastic effect on the way functional requirements are translated into the structure of application code. However, after decades of experience with OOP, it has been shown that a strict separation of all concerns is not feasible as there are requirements that are the same across all components of a system. Examples for this include but are not limited to transaction management, logging as well as enforcing security policies [26]. Concerns like these are equally relevant to several components while at the same time not capurable as separate components in traditional OOP. As they spread across several components, they are often referred to as cross-cutting concerns. AOP is used for central encapsulation of cross-cutting concerns into so-called aspects, which avoids the scattering of duplicated code for realizing them across the codebase. The difference of the two approaches is shown in Figure 9.

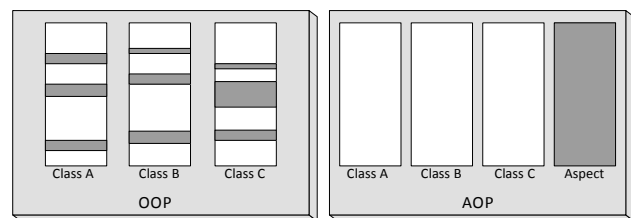


Figure 9. Using AOP to avoid duplication of cross-cutting concerns.

Utilizing DI and AOP, Spring Security allows central definition of security constraints. By integrating with the hosting web container, a central hook is implemented by which a chain of filters can monitor and control the processing of HTTP requests as well as the execution of application components. This enables Spring Security to capture all elements of an application’s architecture while thoroughly ensuring its security requirements using a declaratively configurable mechanism. Figure 10 visualizes how a request is processed.

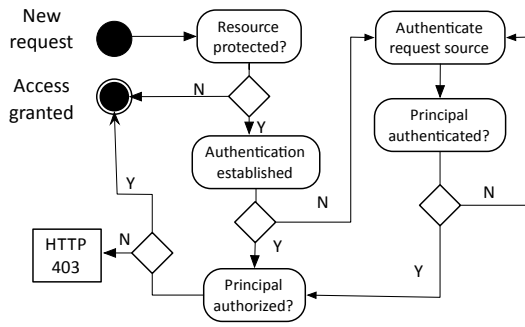


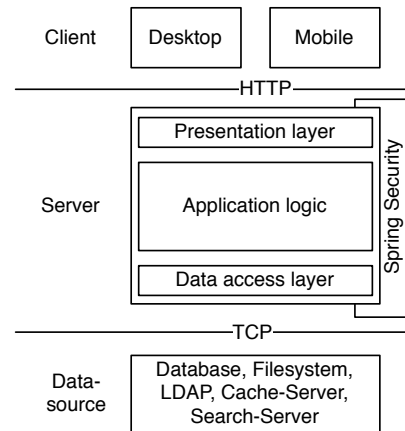
Figure 10. Overview of request processing within Spring Security.

This central hook determines whether an incoming request is trying to access a protected resource according to the supplied configuration. If this is the case, the "AuthenticationManager" (AM) is requested to authenticate and return the current principal, an abstract notion for, e. g., the currently logged in user, which is used by the "AccessDecisionManager" (ADM) to determine whether its role has the permission required for the protected resource in question.

These two components can be controlled in a very flexible manner. For instance, during authentication, the AM can be configured to consult different providers, which in turn compare the principal's credentials by querying relational databases, LDAP directories or Single-Sign-On authentication servers. The ADM can be controlled by assigning static key/value-pairs of resources and required permissions or by enabling the dynamic execution of AOP-driven components. As the flow of the request processing shows, no access is granted unless the source of the request is properly authenticated and the principal possesses sufficient permissions. This effectively realizes the "Fail Safe Defaults" principle as it will return a security error unless both conditions are met. The following shows how Spring Security can be used to enforce compliance with the static and dynamic security requirements as specified in the permission- and constraint catalog in Figure 5.

### B. Static aspects of security

Now that the basic functionality and main components of Spring Security were introduced, this section will focus on how the AM and ADM are used to implement the security requirements stemming from the SDRE role concept defined in chapter V. The following Figure 11 visualizes the architecture of *MinaBASE*. As shown, the application serves multiple types of clients while also utilizing heterogeneous datasources. It is structured using the popular 3-tier architecture, which divides the application into presentation logic for request processing, application logic for satisfying business requirements and components for accessing the various datasources.

Figure 11. Schematic overview of *MinaBASE* architecture.

The following will show how Spring Security is used to implement the security requirements across the entire application architecture in a coherent way. Extending the security mechanisms requires the ADM to use a *FilterSecurityInterceptor* (FSI) for securing the presentation layer as well as a *MethodSecurityInterceptor* (MSI) for the application layer. To protect the application layer, a configuration of the MSI is required that determines which permissions the role of the current principal must possess to invoke components for data access and application logic. This can be realized by placing annotations directly in the application source code or through a central AOP configuration. The latter variant is used due to easier maintenance and therefore shown in Listing 1. For the protection of method invocations, a so-called pointcut, which is an entry point for the execution of code formulated as AOP-advice, is associated with a permission, whose presence will be checked by the MSI.

```
<global-method-security>
<protect-pointcut
  expression="execution(*
    edu.kit.minabase.*CompetenceDAO.get*(..))"
  access="PERM_R_Competence"/>
<protect-pointcut
  expression="execution(*
    edu.kit.minabase.*CompetenceDAO.save(..))"
  access="PERM_W_Competence"/>
<protect-pointcut
  expression="execution(*
    edu.kit.minabase.*CompetenceDAO.delete(..))"
  access="PERM_W_Competence"/>
</global-method-security>
```

Listing 1. Configuration of the *MethodSecurityInterceptor*.

This restricts the data access to competences by requiring the presence of the "PERM\_R\_Competence" permission for the execution of methods, whose names start with "get" and are located within the *CompetenceDAO* class. On top of that the methods to insert, update or delete a competence requires the "PERM\_W\_Competence"-permission to be assigned to the role of the current user. The assignments of permissions



to roles can be altered using an administrative interface at runtime. The invocation of methods for modifying and deleting other information entities within *MinaBASE* as well as the execution of application logic can be restricted in a similar fashion. To make sure that the pointcut expressions are executed as intended, the following Listing 2 shows how their effectiveness can be tested using standard JUnit tests.

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations={
    "file:src/spring-test.xml"})
public class CompetenceDAOSecurityTests {
    @Autowired
    CompetenceDAO dao;

    @Test(expected = AuthenticationException.class)
    public void unauthenticated_get(){
        dao.get(1L);
    }

    @Test
    public void authenticated_get(){
        login("user", "user");
        Competence c = dao.get(1L);
        assertNotNull(c);
    }

    @Test(expected = AccessDeniedException.class)
    public void insufficient_permissions_save(){
        login("user", "user");
        Competence c = new Competence();
        dao.save(c);
    }

    @Test(expected = AccessDeniedException.class)
    public void insufficient_permissions_delete(){
        login("user", "user");
        dao.delete(1L);
    }

    @Test
    public void sufficient_permissions(){
        login("admin", "admin");
        Competence c = new Competence();
        dao.save(c);
        dao.delete(1L);
    }

    private void login(String u, String pw) {
        UsernamePasswordAuthenticationToken token =
            new UsernamePasswordAuthenticationToken(u, pw);
        SecurityContextHolder.getContext()
            .setAuthentication(token);
    }
}
```

Listing 2. Testing effectiveness of pointcut expressions.

The annotations on the class are necessary to specify the JUnit-Test-Runner as well as to provide the location of the configuration file to the Spring Framework. Given this information, Spring will bootstrap the DI-container in the background, inject the dependency *CompetenceDAO* into the test class and run all methods marked with the JUnit `@Test`-annotation. The first testcase simulates anonymous access and makes sure that no unauthenticated user can run the `get`-method of the *CompetenceDAO* component

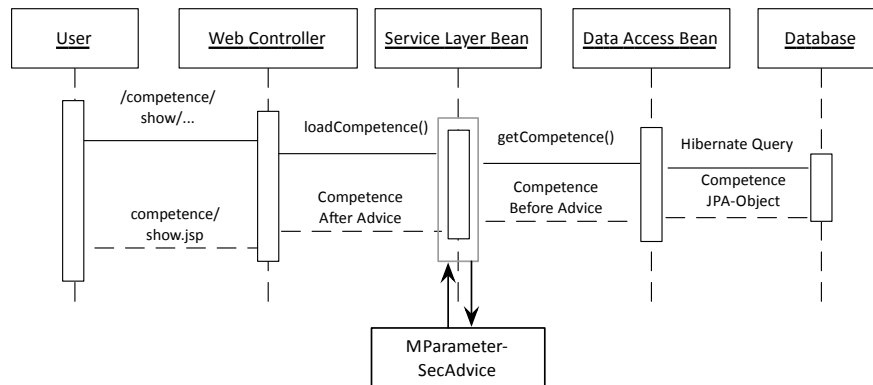
by expecting the test code to raise an exception of type `AuthenticationException`, which is part of Spring Security. If this exception is not raised, the testcase is considered to have failed by JUnit and will be reported as such. The next testcase is responsible for proving that authenticated users can execute the `get`-method without causing an exception. The credentials specified in the `login`-method refer to an In-memory *AuthenticationProvider*, which is used during the tests only. The configuration is shown in the following Listing 3.

```
<sec:authentication-manager>
  <sec:authentication-provider>
    <sec:user-service>
      <sec:user name="user" password="user"
        authorities="PERM_R_COMPETENCE" />
      <sec:user name="admin" password="admin"
        authorities=
          "PERM_R_COMPETENCE, PERM_W_COMPETENCE" />
    </sec:user-service>
  </sec:authentication-provider>
</sec:authentication-manager>
```

Listing 3. In-Memory AuthenticationManager.

By placing this into the test-specific Spring configuration, the runtime duration of the tests can be reduced as no database lookups are required. For reasons of clarity, the permissions are not grouped to roles here, but are added directly to the users of the *In-Memory AuthenticationProvider*. In a production scenario the authentication process is backed by a chain of *AuthenticationProviders* which consult different datasources (LDAP-repositories, a relational database or the filesystem). But to minimize the dependencies on the surrounding environment, the In-Memory approach was chosen, as it enables the tests to be run from anywhere. As the configuration shows, the first user only possesses the `PERM_R_Competence` permissions, while the `admin`-user additionally owns `PERM_W_Competence`. The is important for the rest of the testcase from Listing 3, as the third testcase tries to run the `save`-method, which has been configured to require the higher permission. To ensure the effectiveness of the pointcut expression in Listing 1, this testcase is expected to raise an `AccessDeniedException` or to fail otherwise. The following testcase assures the same for the `delete`-method. The final testcase is used to prove that no exception is raised when a user with sufficient permissions runs both methods. While the pointcuts in Listing 1 are expressed in a very generic and broad way, these testcases will assure their effectiveness. By minimizing dependencies to external datasources, this way of testing can be used to prove the fulfillment of the security requirements on the level of application logic and data access.

For securing the presentation layer, combinations of URL-patterns for protected regions and required permissions are specified, which are evaluated by the FSI during the monitoring of HTTP request processing. An excerpt of the necessary configuration is shown in Listing 4.

Figure 12. Integration of the MParameterSecAdvice in *MinaBASE*.

```

<http auto-config="false"
  access-denied-page="/denied.jsp">
  <intercept-url
    pattern="/static/*.*" filters="none"/>
  <intercept-url
    pattern="/competence/show/**"
    access="PERM_R_Compotence"/>
  <intercept-url
    pattern="/competence/edit/**"
    access="PERM_W_Compotence"/>
  <form-login login-page="/login.jsp"
    authentication-failure-url="/login.jsp?error=1"/>
  <logout logout-success-url="/logout.jsp"/>
</http>

```

Listing 4. Configuration of the FilterSecurityInterceptor.

Due to the URL-patterns being evaluated from top to bottom, the monitoring is at first disabled for static resources to achieve higher performance. Thereafter, permissions for visiting URLs matching the location for display and editing of competences are stated. The final step is the declaration of URLs, to which the AM will redirect unauthenticated users, that try to access a protected resource as well as URLs for authentication failures and the termination of a user's session. These settings ensure that protected areas are not reachable for users that don't possess the required permissions. To improve the user experience, links to sections the user does not have access to should not be displayed in the first place. To achieve this, the generation of HTML needs to be controlled with permissions in mind. Spring Security is bundled with an extension that allows fragments of Java ServerPages (JSP) to be rendered according to the current user's permissions, which is demonstrated in Listing 5.

```

<sec:authorize ifAllGranted="PERM_W_Compotence">
  <a href="/competence/edit/...">
    Edit this competence</a>
</sec:authorize>

```

Listing 5. Permission based generation of the user interface.

This JSP-Tag assures that links to the area for editing competences are only rendered to those users that have the "PERM\_W\_Compotence" permission. These three listings show how Spring Security can be used to employ a homogeneous system of permissions that stems from the SDRE permission catalog and covers the entire application architecture from data access to the presentation layer. At runtime these permissions are assigned to roles from the designed RBAC model.

### C. Dynamic security aspects

In the previous section, security aspects were considered, which could be fulfilled by statically restricting access to a protected resource by requiring a specific permission to be held by the current principal. While most aspects of the permission catalog are covered by this approach, entries of the constraint-catalog as depicted in Figure 5 cannot be implemented in this fashion, because of their dynamic nature, which means, these constraints cannot be enforced at build-time, but only at runtime. As an example, the filtering of fabrication-specific TP of a competence's detailed view is used. To avoid code duplication whenever fabrication-specific TP of a *MinaBASE* information entity shall be filtered, this concern is encapsulated into a separate AOP-Advice called "MParameterSecAdvice", whose integration into the method's call flow is illustrated in Figure 12.

A request for the detailed view of a competence is received by a Web-controller, which initiates the data access for the current competence by invoking methods from the service layer. Once this competence is loaded as a database object, the MParameterSecAdvice is hooked into the execution flow using AOP-Weaving. The job of this component is to iterate over the competence's parameter collection and filter out those parameters to which the current principal has no permission. The revised competence object is then

returned to the controller which starts the generation of HTML templates and sends the result to the browser.

After explaining the implementation from a high-level point of view, the following will focus on the detailed implementation by describing the source code of the MParameterSecAdvice, which is shown in Listing 6.

```
public class MParameterSecAdvice {
    public void injectAfter(Object ret) {
        SecurityContext ctx =
            SecurityContextHolder.getContext();
        Authentication auth = null;
        boolean fp_permission = false;
        HasAttributes c =
            (HasAttributes) ret;
        Set<CAttribute> cattrs
            = c.getAttributes();
        Set<CAttribute> onlyPPAttrs
            = new HashSet<CAttribute>();

        if (ctx.getAuthentication() != null){
            auth = ctx.getAuthentication();
            GrantedAuthority[] permissions =
                auth.getAuthorities();
            for (int i = 0; i < permissions.length; i++){
                String perm
                    = permissions[i].getAuthority();
                if (perm.equals(Constants.FP_PERM)){
                    fp_permission = true;
                    break;
                }
            }
        }

        if (fp_permission == false){
            if (cattrs != null){
                for (CAttribute current : cattrs){
                    if (isFP(current) == false){
                        onlyPPAttrs.add(current);
                    }
                }
            }
            c.setAttributes(onlyPPAttrs);
        }

        private boolean isFP(CAttribute a){
            Attribute type = a.getAttribute().getType();
            return type.getId().equals(Constants.FP_ID);
        }
    }
}
```

Listing 6. Parameter filtering constraint within MParameterSecAdvice.

At first, the SecurityContext is retrieved, which contains all information about the current principal. If the current principal can be fetched, its permissions are loaded by invoking the "getAuthorities"-method from the current Authentication object. The next step is iterating over the list of permissions to find out, whether the current principal has the authority to view the fabrication-related competence parameters. If this is the case, then filtering of parameters can be skipped later on. If this is not the case, the parameters need to be filtered. If the current principal does not possess the required permission, a new collection of parameters is built by iterating over all the parameters of the current

competence and inserting only product-related parameters into it. Afterwards the competence is updated by setting the new collection of product-related parameters as the competence's attributes. As the call flow in Figure 12 shows, only the final step, namely rendering of the user interface using HTML templates, is left. As the competence now only consists of product-related parameters, the fabrication-related parameters cannot be rendered to the user. The following Listing 7 shows how the advice is weaved into the execution using a pointcut expression.

```
<bean id="competenceDAO"
    class="edu.kit.minabase.data.CompetenceDAO"/>
<bean id="mParamSecAdvice"
    class="edu.kit.minabase.aop.MParameterSecAdvice"/>

<aop:config>
    <aop:aspect ref="mParamSecAdvice">
        <aop:pointcut id="afterDaoPointcut"
            expression="
                execution(*
                    edu.kit.minabase.*
                    .CompetenceDAO.*getAttributes(..))"/>
        <aop:after-returning returning="ret"
            method="injectAfter"
            pointcut-ref="afterDaoPointcut"/>
    </aop:aspect>
</aop:config>
```

Listing 7. Pointcut expression for the MParameterSecAdvice.

At first the beans "competenceDAO" and the "mParamSecAdvice" are declared to the DI container. Subsequently, the pointcut expression states where the advice is to be executed. In our concrete example this expression refers to the "getAttributes"-method of the CompetenceDAO. As advices can run before, during or after the method referenced in the pointcut expression, the last step is to state when the advice should run. Due to the fact that we need to have the competences populated with all of its parameters, we need to run the advice after the data access code of the CompetenceDAO class. Using the "returning"-attribute, we specify the name of the method parameter that will be used to transit the return value of the data access code to the advice. The value "ret" corresponds to the method parameter of the "injectAfter"-method inside of MParameterSecAdvice as can be seen in Listing 6. Inside this method, the "ret"-variable is assigned and type casted to an interface type called "HasAttributes", which is an interface that all information entities within *MinaBASE* implement whenever they can be characterized by parameters. Using this interface instead of a concrete class make this advice useful to all those entities as the implementation is not bound to a single one of them. The advantage of this approach is the fact that the permission based filtering is encapsulated into the MParameterSecAdvice once and can be applied declaratively to multiple application components without code duplication and mixture of concerns by simple configuration in a similar fashion as described in Listing 7.

## VII. RELATED WORK

The significance of RE activities when implementing RBAC has led to the development of various RE methodologies. These can be classified as top-down, bottom-up and hybrid approaches. To support enterprise-wide RE, Role Mining has been used to automate parts of the RE activities. In this section, we will discuss these approaches.

Top-down approaches use abstract concepts like work profiles or business functions as a starting point. These are decomposed into smaller parts and mapped onto permissions which enable an aggregation to roles. As an example, in [27] Roeckle et al. define a formal, process-oriented approach for role finding combining an RBAC metamodel with a procedural model for interfacing business processes in order to automatically derive roles from a process view using a tool called "RoleFinder". In [19] Coyne employs user activities to also identify roles in a top-down manner. Permissions are allocated to roles using the Principle of Least Privilege, as only those permissions are assigned which are necessary to complete a role. Constraints are defined and role hierarchies are built subsequently. Fernandez and Hawkins introduce a semi-formal approach based on textual description of system and user interaction utilizing use-cases [28]. By extending use-cases with rights specification in the form of actors, activity descriptions, preconditions, exceptions and postconditions, all roles and permissions necessary for a system can be determined.

Bottom-up approaches collect permissions as pairs of operations on resources within information systems and use these as a building block for role aggregation using business functions. In [29] Thomsen et al. introduce seven abstract layers to facilitate security management based on RBAC. These layers enable the identification of permissions from objects as well as associated methods and roles for usage by security administration and application developers. Epstein and Sandhu propose the use of the Unified Modeling Language to document each layer introduced by Thomsen et al. in [30].

Hybrid approaches try to combine both techniques as described above by parallelizing the RE activities or by basing them on an iterative-incremental process [31]. As an example, Epstein and Sandhu propose a conceptual framework in [32] to derive roles in a top-down and bottom-up manner.

RE is useful when the quality of documentation is high and if the amount of tasks, work-profiles or business processes is manageable. In case of dozens of processes, thousands of resources and permissions, a proper decomposition and aggregation to role concepts becomes rather difficult. These conditions have led to Role Mining approaches, in which tools and algorithms from data mining, such as clustering and neural networks are used to derive potential roles automatically [33]. In [34] Fuchs and Pernul introduce

HyDRo, which is a tool-supported methodology that facilitates the definition of enterprise-wide roles by combining elements from RE and Role Mining.

As described in detail in section IV, the underlying methodology of this paper is SDRE [4]. As scenarios are used as a building block to derive complete work profiles and associated tasks, but also a mapping of tasks to permissions result in aggregated roles, SDRE - even with our adaptations to it - can be characterized as a hybrid approach. For RE in the context of collaborative knowledge management systems, an alternative configuration of the SDRE mechanism has been used throughout this paper. The approach has been applied with special focus on the use of *MinaBASE* in business processes of MST enterprises and cooperation networks. Instead of using scenarios as the main input, work areas within the business processes of an MST-company are examined, whether they include tasks in which *MinaBASE* can be used to increase added value. To these tasks scenarios are assigned in order to determine which authorizations are needed to fulfill them. Based on this information, a role concept can be derived and further refined. As we only adapt the input variables to the methodology, basic principle of SDRE is preserved, while better results for the creation of the role concept in the context of collaborative knowledge management systems are expected as the adapted methodology is closer to the business processes of a MST-company.

## VIII. CONCLUSION

In this paper, a role concept for the process knowledge database *MinaBASE* has been developed based on a systematic methodology called Scenario-driven role engineering. The implementation of this role concept within an IoC-Framework such as Spring has been demonstrated by utilizing Spring Security and technologies such as AOP. At first, the *MinaBASE* approach, mechanisms for access control with a special focus on role based access control as well as the Scenario-driven role engineering methodology were introduced. Following this, careful adjustments were made to the inputs of the SDRE process resulting from the background and purpose of *MinaBASE* without hurting the methodology's idea and principles. Then the application of the SDRE process was shown including examples on how to derive a minimal set of permissions enabling each role to fulfill its work profile. In the following section the implementation of the derived role concept using Spring Security is described in detail. Important concepts are Dependency Injection and AOP, as they enable Spring Security to ensure static and dynamic security requirements across the entire application architecture. For the implementation of security requirements that can be decided at runtime only, an example was given in order to prevent the disclosure of fabrication-specific parameters for non-authorized persons.

## REFERENCES

- [1] D. Kimmig, A. Schmidt, K. Bittner, and M. Dickerhof, "Application of Scenario-driven Role Engineering to the MinaBASE Process Knowledge Database," in *SECURWARE 2011, Proceedings of the Fifth International Conference on Emerging Security Information, Systems and Technologies*. 978-1-61208-146-5, 2011, pp. 125–132.
- [2] U. Hansen, C. Germer, S. Büttgenbach, and H. Franke, "Rule based validation of processing sequences," in *Techn. Proc. MSM*, 2002.
- [3] D. F. Ferraiolo and R. Kuhn, "Role-based access control," in *Proceedings of 15th NIST-NCSC National Computer Security Conference*, October 1992, pp. 554–563.
- [4] G. Neumann and M. Strembeck, "A scenario-driven role engineering process for functional RBAC roles," in *SACMAT '02: Proceedings of the seventh ACM symposium on Access control models and technologies*. New York, NY, USA: ACM Press, 2002, pp. 33–42.
- [5] I. Nonaka and H. Takeuchi, *The knowledge-creating company: How Japanese companies create the dynamics of innovation*. Oxford University Press, USA, 1995.
- [6] M. Dickerhof, "Prozesswissensmanagement für die Mikrosystemtechnik." 2003.
- [7] M. Dickerhof and A. Parusel, "Bridging the Gap—from Process Related Documentation to an Integrated Process and Application Knowledge Management in Micro Systems Technology," *Micro-Assembly Technologies and Applications*, vol. 260, pp. 109–119, 2008.
- [8] M. Dickerhof, O. Kusche, D. Kimmig, and A. Schmidt, "An ontology-based approach to supporting development and production of microsystems," *Proc. of the 4th Internat. Conf. on Web Information Systems and Technologies*, 2008.
- [9] D. Kimmig, A. Schmidt, K. Bittner, and M. Dickerhof, "Modeling of Microsystems Production Processes for the MinaBASE Process Knowledge Database Using Semantic Technologies," in *Proc. of the The 3rd Internat. Conf. on Information, Process, and Knowledge Management*, 2011, pp. 17–23.
- [10] R. C. David F. Ferraiolo, D. Richard Kuhn, *Role-Based Access Control*, 1st ed., ser. Computer Security Series. Artech House, Inc., 2003.
- [11] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed nist standard for role-based access control," *ACM Transactions on Information and System Security (TISSEC)*, vol. 4, no. 3, pp. 224–274, 2001.
- [12] E. Glaser, "A brief description of privacy measures in the Multics operating system," in *Proceedings of AFIPS SJCC*, vol. 30. Montvale, N.J.: AFIPS Press, 1967.
- [13] M. Benantar, *Access Control Systems - Security, Identity Management and Trust Models*, 1st ed. Springer Science+Business Media, Inc., 2006.
- [14] M. D. S. Jerome H. Saltzer, "The protection of information in computer systems," in *Proceedings of fourth ACM Symposium on Operating System Principles*, 1975.
- [15] *Trusted Computer System Evaluation Criteria - DoD 5200.28 Std.* Department of Defense, 1985.
- [16] D. E. Bell and L. J. LaPadula, "Secure computer systems: Mathematical foundations," Mitre Corporation Bedford, Tech. Rep., March 1973.
- [17] K. Biba, "Integrity considerations for secure computer systems," Mitre Corporation Bedford, Tech. Rep., April 1977.
- [18] D. F. Ferraiolo, J. A. Cugini, and D. R. Kuhn, "Role Based Access Control (RBAC): Features and Motivations," in *In Proceedings of 11th Annual Computer Security Application Conference*. IEEE Computer Society Press, 1995, pp. 241–48.
- [19] E. J. Coyne, "Role Engineering," in *RBAC '95: Proceedings of the first ACM Workshop on Role-based access control*. New York, NY, USA: ACM Press, 1996, p. 4.
- [20] HL7 Security Technical Committee, "HL7 Role Based Access Control (RBAC) Role Engineering Process," January 2005.
- [21] K. Boehm, W. Engelbach, J. Härtwig, M. Wilcken, and M. Delp, "Modelling and implementing pre-built information spaces. architecture and methods for process oriented knowledge management," *Journal of Universal Computer Science*, vol. 11, no. 4, pp. 605–633, 2005.
- [22] B. Alex and L. Taylor, "Spring Security Reference Documentation," URL: <http://static.springsource.org/spring-security/site/docs/3.1.x/reference/springsecurity-single.html>, accessed: 2012-07-12.
- [23] M. Fowler, "Inversion of Control Containers and the Dependency Injection pattern," URL: <http://martinfowler.com/articles/injection.html>, Januar 2004, accessed: 2012-07-12.
- [24] R. Johnson and J. Hoeller, *J2EE Development without EJB*, 1st ed., ser. Expert on-on-one. Wiley Publishing, Inc., 2004.
- [25] E. Dijkstra, *A discipline of programming*. Englewood Cliffs, NJ: Prentice Hall, 1976.
- [26] O. Böhme, *Aspektorientierte Programmierung mit AspectJ 5*. dpunkt.verlag, 2006.
- [27] H. Roeckle, G. Schimpf, and R. Weidinger, "Process-oriented approach for role-finding to implement role-based security administration in a large industrial organization," in *Proceedings of the fifth ACM workshop on Role-based access control*, ser. RBAC '00. New York, NY, USA: ACM, 2000, pp. 103–110.
- [28] E. B. Fernandez and J. C. Hawkins, "Determining role rights from use cases," in *Proceedings of the second ACM workshop on Role-based access control*, ser. RBAC '97. New York, NY, USA: ACM, 1997, pp. 121–125.

- [29] D. Thomsen, D. O'Brien, and J. Bogle, "Role based access control framework for network enterprises," in *Computer Security Applications Conference, 1998, Proceedings., 14th Annual*, Dec 1998, pp. 50–58.
- [30] P. Epstein and R. Sandhu, "Towards a UML based approach to role engineering," in *Proceedings of the fourth ACM workshop on Role-based access control*, ser. RBAC '99. New York, NY, USA: ACM, 1999, pp. 135–143.
- [31] A. Kern, M. Kuhlmann, A. Schaad, and J. Moffett, "Observations on the role life-cycle in the context of enterprise security management," in *Proceedings of the seventh ACM symposium on Access control models and technologies*, ser. SACMAT '02. New York, NY, USA: ACM, 2002, pp. 43–51.
- [32] P. Epstein and R. Sandhu, "Engineering of Role/Permission Assignments," in *Proceedings of the 17th Annual Computer Security Applications Conference*, ser. ACSAC '01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 127–.
- [33] J. Vaidya, V. Atluri, and Q. Guo, "The role mining problem: finding a minimal descriptive set of roles," in *Proceedings of the 12th ACM symposium on Access control models and technologies*, ser. SACMAT '07. New York, NY, USA: ACM, 2007, pp. 175–184.
- [34] L. Fuchs and G. Pernul, "HyDRo – Hybrid Development of Roles," in *Information Systems Security*, ser. Lecture Notes in Computer Science, R. Sekar and A. Pujari, Eds. Springer Berlin / Heidelberg, 2008, vol. 5352, pp. 287–302.

# A Scalability Analysis of an Architecture for Countering Network-Centric Insider Threats

Faisal M. Sibai  
Volgenau School of Engineering  
George Mason University  
Fairfax, VA 22030, USA  
Email: fsibai@gmu.edu

Daniel A. Menascé  
Dept. of Computer Science, MS 4A5,  
George Mason University  
Fairfax, VA 22030, USA  
Email: menasce@gmu.edu

**Abstract**—Dealing with the insider threat in networked environments poses many challenges. Privileged users have great power over the systems they own in organizations. To mitigate the potential threat posed by insiders, we introduced in previous work a preliminary architecture for the Autonomic Violation Prevention System (AVPS), which is designed to self-protect applications from disgruntled privileged users via the network. We also provided insight on an architecture extension and how well the AVPS can scale. This paper extends the scalability model of our previous work and presents additional results. We conducted a series of experiments to assess the performance of the AVPS system on three different application environments: File Transfer Protocol (FTP), database, and Web servers. Our experimental results indicate that the AVPS introduces a very low overhead despite the fact that it is deployed in-line. We also developed an analytic queuing model to analyze the scalability of the AVPS framework as a function of the workload intensity. We show model results for a varying number of applications, users, and AVPS engines.

**Keywords**- insider threat, scalability, network security.

## I. INTRODUCTION

Defeating the insider threat is a very challenging problem in general. An insider is a trusted person that has escalated privileges typically assigned to system, network, and database administrators; these users usually have full access and can do almost anything to the systems and applications they own. Users with escalated privileges within an organization are trusted to deal with and operate applications under their control. This trust might be misplaced and incorrectly given to such users. It is extremely difficult to control, track or validate administrators and privileged user actions once these users are given full ownership of a system. The recent disclosure by Wikileaks of U.S. classified embassy foreign policy cable records provides a perfect example of an insider attack [1] [2]. In this disclosure, an insider with unfettered access to data at his classification level was able to access data over a secure network using laptops that had functional DVD writers. Our approach to mitigate the insider threat allows for users or groups of users to be treated differently despite having the same classification level [3]. The approach limits and controls network access through an in-line component that checks access to specific applications based on policies that can be as specific or granular as needed.

In our prior work, we introduced a framework that self-protects networks in order to mitigate the insider threat [3]. The framework, called AVPS (Autonomic Violation Prevention System), controls and limits the capabilities provided to administrators and privileged users in organizations. AVPS concentrates entirely on detecting and preventing usage policy violations instead of dealing with viruses, malware, exploits, and well-known intrusions. In our implementation, the AVPS monitors events and takes actions for conditions that occur, as specified by Event-Condition-Action (ECA) commonly used in security-centric systems and autonomic computing [4].

Our most recent work [1] significantly extends our earlier work [3] and presents a scalable AVPS architecture and supports its design with experimental results and a theoretical queuing modeling. We presented the results of experimental evaluations of the AVPS architectures as well as the analysis of its performance overhead on three different types of application servers: FTP, database, and web server. We specifically measured the average throughput, average transfer time, average CPU utilization, and provided 95% confidence intervals for all three measurements. We also used a queuing theoretic analytic model to predict the scalability of the AVPS for different workload intensity values for these three types of applications. It is also worth noting that the previous design of the AVPS architecture considered scalability, manageability, application integration, ease of use, and the enforcement of separation of duties. This paper extends our previous work [1] in that it presents extra scalability cases where application, users, and the number of AVPS engines vary. We present an architecture and an explanation for each case.

There has been prior work in this area at the application, host, and network levels [5] [6] [7] [8] [9]. The previous methods have applied self-protecting capabilities by either considering single applications on the host or more towards vulnerabilities, malware, exploits and traditional threats.

The rest of the paper is organized as follows. Section II discusses related work. Section III presents some of the major challenges and requirements faced in the design of AVPS. The next section presents a scalable architecture for the AVPS framework. Section V presents an experimental evaluation and a thorough performance and scalability analysis of AVPS for all three different cases. Finally, Section VI presents the

conclusion, final remarks, and future work.

## II. RELATED WORK

There has been substantial work in scalability analysis and performance enhancement of network security. We discuss specifically some of the major work related to intrusion prevention systems, firewalls and Snort respectively.

In [10], the authors create a framework that can enhance inline intrusion prevention systems performance by utilizing future commodity hardware to the fullest. In [11] the authors of the NIST SP800 stress on scalability as an extremely important part of any deployment of inline intrusion prevention system to be successful. In [12], the authors design a network intrusion prevention system that combines the use of software-based NIPS and a network board processor. Their focus on a method for boosting system performance resulted in a 45% improvement in performance allowing speeds to reach 1Gbit/s. In [13], the authors presented a system called Gsnort that utilizes a GPU to offload pattern matching computations. The system was able to achieve a maximum throughput of 2.3 Gbit/s, in a real world scenario and outperformed conventional Snort by a factor of two. The authors in [14] point out some challenges and scalability issues that might arise when it comes to intrusion detection systems. In [15] the authors present "Para-Snort, a structure for a multithreaded Snort for high performance Network Intrusion Detection Systems and anti-virus on a multi-core IA; they also analyze the performance impact of load balancing and multi-pattern matching.

On the firewall side, the authors of [16] implemented a scalable packet classification architecture resulting in a firewall that achieves a classification throughput of 50 million packets/s. The authors in [17] present a fast and highly scalable approach for discovering anomalies in firewall policies and resolving them. The results of their heuristic algorithm achieved from 40% to 87% improvement in the number of comparisons overhead.

The authors in [18] designed and tested a multithreaded Snort that uses flow pinning as a major optimization technique to improve Snort performance and achieve significant speedups. In [19], the authors present a mechanism to split traffic into different Snort sensors; the system is adaptive and is able to adjust the splitting of policies in order to reduce load disparity among sensors. The authors of [20] compared the performance and accuracy of Suricata and Snort and showed that Snort had a lower system overhead than Suricata utilizing a single core. At the same time, Suricata indicated that it was more accurate in the environment where multi-cores were available. In [21], the authors compared the performance of Snort NIDS under both windows 2003 and Linux and showed that Snort used on a Linux machine with a small NAPI (New API) budget would yield a substantial performance gain for Snort over Windows under all different malicious traffic loads. On the other hand, Windows showed better performance for Snort under moderate normal traffic load conditions.

## III. CHALLENGES AND REQUIREMENTS

The following major challenges play a primary role in the success of the AVPS framework: scalability in production environments, support for encrypted network traffic, integration with multiple types of application servers on the network, and ease of deployment in large production environments. This paper mainly addresses scalability and performance issues and sheds some light on all four challenges. Security mechanisms usually pose additional demands on system resources and may compromise system performance. In some cases, the use of security mechanisms has been abandoned due to the need to run systems efficiently. Thus, it is important to understand security-performance tradeoffs [22].

Scalability is an absolute requirement for production environments. The AVPS solution is an in-line solution that intercepts every single packet that traverses the local area network that is destined to an application server. Therefore, it could become a focal point and a possible bottleneck. The primary goal of our solution is to scale with growing network and application demands. The AVPS architecture should allow for horizontal scaling to cope with high-volume environments. This requirement is further discussed in more detail in the following sections.

Encryption is another important challenge in the design of our solution. SSH and SSL are widely used in local area networks for information retrieval and administration of applications and devices. The AVPS performs packet inspection on some or all (depending on the application) packets that pass through it. This poses a challenge that is handled in our solution through one of the following methods: (1) decrypting the traffic that passes through the AVPS and then re-encrypting it for delivery to its destination using `viewSSLd` [23] or `netintercept` [24] for example, (2) completely off-loading the encryption/decryption requirements to external hardware-based devices that sit before and after the AVPS, or (3) decrypt the traffic by having a legitimate man-in-the-middle host that decrypts and re-encrypts the traffic and delivers it to the destination [25]. This paper does not discuss encryption in any further detail.

Application server integration is also extremely important. With the wide range of applications deployed in production environments, the AVPS framework must be capable of interpreting and understanding requests and responses that it intercepts. The AVPS is based on intercepting, not necessarily inspecting, every single packet initiated by a host that is delivered from and to an application. This makes application integration completely possible and achievable. Policies deployed on the AVPS are customizable to the desired granularity level and types of attributes (e.g., from very generic, such as IP or user level, to very specific, such as IP, user, application type, request, and response). Thus, it is completely up to the AVPS owner to specify the granularity of what should be inspected and what should be ignored.

Finally, the successful deployment of AVPS in large environments is crucial. The AVPS solution should be easy



to deploy and maintain and should be capable of handling heavy traffic loads. Current environments have hundreds if not thousands of servers with networks that are capable of handling and processing 100 to 1000 Mbps of traffic. A solution that handles thousands of servers through a handful of clustered AVPS compute nodes is part of the architecture discussed in the remaining sections of this paper.

#### IV. SCALABLE AVPS ARCHITECTURE

For the AVPS to achieve its goal of solving the insider threat problem, it must be placed in-line between clients and internal application servers. This way, the AVPS is capable of intercepting every single packet that flows from clients to applications and back in order to take the correct actions when a rule in a policy is matched.

##### A. The AVPS Architecture

Figure 1 depicts the architecture of the AVPS framework. Performance and high availability are extremely important since the AVPS is located between the clients and the application servers. Traffic coming from a pool of  $M$  clients goes through a load balancer that handles incoming requests. The load balancer forwards the traffic to one of  $N$  AVPS engines that process and inspect the incoming traffic. The AVPS engines compare traffic policies that contain rules and actions on how to handle traffic. The policies are stored on a database/multiple databases local to the AVPS engine or on an external database shared by all AVPS engines. Events are stored on a centralized database or multiple databases. Actions are taken on traffic once a rule in a policy has been matched. Examples of possible AVPS actions include dropping, blocking, or replacing traffic as it traverses the engine on its way to application servers. Let there be  $K$  different types of applications servers (e.g., FTP server, database server, Web server).

Figure 2 depicts a flowchart that shows the traffic processing steps taken by the AVPS engine. Traffic is first collected by the machine that runs the AVPS engine. Then, traffic is received by a layer 2 bridge that is responsible for handling incoming and outgoing traffic. The layer 2 bridge flow traffic contains layers 2 and 3 traffic for processing.

Traffic is then forwarded to the normalization and processing module where packets are broken down into pieces that can be matched against rules. Traffic is then matched against policies and rules that are pre-loaded into memory. If there is a rule match, an event or action is generated. If an event or action occurred, it is logged into a database. If the traffic results in an unauthorized action, the traffic will be dropped, blocked or replaced. If the action is authorized, the system starts the cycle again from the traffic collection process. If the process is terminated, the system halts and does not perform any further action.

##### B. Advantage of Using the AVPS

As an example of the advantage of using the AVPS architecture, consider a scenario with multiple database servers scattered over a large geographically distributed network. Assume

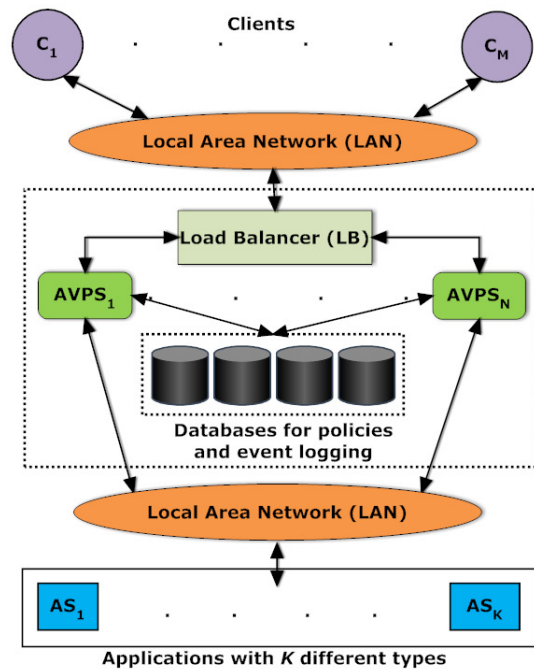


Fig. 1: Architecture of the AVPS framework.

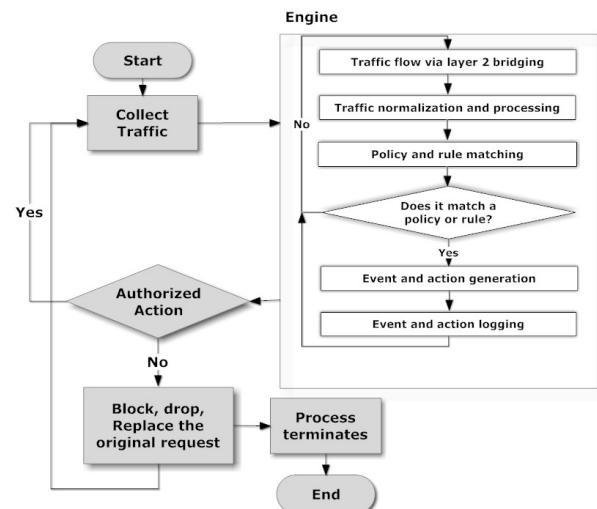


Fig. 2: Steps of the AVPS engine.

that a *top secret* table is replicated in every database server and that we want to have fine access control to this table. Using conventional access control methods, we would be able to limit specific users or roles from accessing the table. This would require manually setting these controls on every database server. This approach has several drawbacks: (1) Manually setting access controls into each server is time consuming and might have a high error rate. (2) This method requires an administrator to know all of the DB servers that live on the network; newly installed DB servers or even covert ones may be missed. (3) The DB owner actually does the changes with no oversight, which contradicts the separation-of-duties concepts. (4) Last but not least, it would be almost impossible

with traditional access control methods to limit access for a specific population of administrators or privileged users, coming from a specific location on the network, accessing the information at a specific time and targeting a specific table. The AVPS would also be a viable solution to better control actions and secure access to and from cloud Infrastructure As a Service (IAAS), Software As a Service (SAAS), and Platform As a Service (PAAS).

In recent work, we showed how the AVPS can automatically generate low level Snort rules from high-level rules [26]. Each high-level rule may generate more than one low level rule. The automatic rule generation takes place offline so that it does not impact performance. The new rules are then automatically loaded into the main memory of the Snort engine used to implement the AVPS. This substantially lowers the amount of time required to manually configure rules and mitigates the drawback mentioned in the example above. In addition to automatic low level rule generation, we used supervised learning (Support Vector Machines (SVM) in our case) to learn new high-level rules [27].

The AVPS is also tamper resistant. It enforces a separation-of-duties policy, i.e., the primary application system owner has no control over the AVPS policies [3]. The AVPS can be deployed to carry insider and regular user traffic or to only carry insider traffic. The proper deployment depends on how the network is setup and on how the network is segmented.

Emerging technologies, such as new network TAPs (e.g., Network Critical V-line TAP [28]), that can handle 1/10 Gbps traffic and allow in-line functionality without introducing a single point of failure, make systems such as the AVPS possible to implement without fault-tolerance concerns.

### C. AVPS vs. Other Solutions

Our prior work [3] distinguishes the AVPS from other systems such as IPS, Firewalls, Host based IPS and Network Admission Control/Network Access Control (NAC). We use Intrusion Prevention Systems (IPS) and Intrusion Detection Systems (IDS) in this paper interchangeably. The only difference between the two is that IDS is considered a passive network monitoring system and IPS is considered an active/inline network monitoring system. Traditional IDS/IPS systems tend to concentrate on users that do not have access to the system and try to exploit, hack, or crack into it. Other enhanced IPS/Firewall systems such as IBM Proventia [29] or Cisco ASA [30] do have enhanced context-aware security but lack insider threat defeating capabilities. The AVPS, on the other hand, is designed with the insider threat in mind. Moreover, as indicated previously, the AVPS uses self-learning techniques to learn high-level rules that are automatically translated into low level Snort rules.

## V. PERFORMANCE ASSESSMENT OF THE AVPS

This section presents an experimental evaluation of the AVPS in a controlled environment. We describe the experimental testbed, analyze the results, and present a scalability analytical model based on the M/M/N/M queuing model [31].

### A. Experimental Testbed

The experiments conducted in this paper measure the impact of a rule that exists in the engine's main memory and is used to match a specific network pattern while the traffic flows in an in-line fashion through the AVPS. While we understand that a growing number of rules in policies may have an overall performance impact, we have not seen this to be an issue in our system when performance profiling [32], fast pattern matching [32], and other Snort [33] [32] [34] tweaks are performed, using third party plug-ins such as Barnyard [35], and the adequate CPU and main memory resources, and number of AVPS engines re available at the time the AVPS solution is deployed. The experiments conducted in this paper do not cover the effects of a growing number of rules over time due to the various factors that need to be considered and tested separately, we plan to conduct further testing for this in the future.

We based our experiments on three different applications: FTP, database, and Web server. The specification of the environment and the experimental testbed is shown in Figure 3.

In this environment, the client requests services from application servers, which respond to the requests. All traffic between client and server is monitored and inspected by the AVPS. A controlling host controls the environment and collects the results of the experiments (see Figure 3).

Apache JMeter 2.4 [36] was used on the client to conduct both FTP and Web experiments. We measured the average throughput and average transfer time in both cases. For the database experiment, mysqlslap [37] was used to measure the average response time.

On the AVPS we used Snort-inline 2.8.6.1 [38]. Snort is highly used in academic IDS/IPS research experiments. Other tools are also used in academic research (e.g., Bro [39] and EMERALD [40]). We used Linux iptables [41], a firewall package installed under RedHat, Fedora, and Ubuntu Linux, in conjunction with Snort in-line to filter packets as they come into the AVPS and leave. We used MySQL 5.1 [42] to store events and event packet captures. We used BASE [43] to query

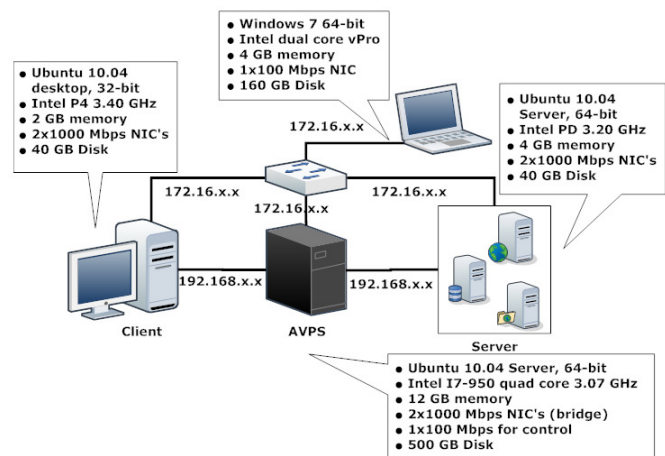


Fig. 3: Experimental environment.

the DB and display the events in the browser.

We configured three different application servers: (1) vsftpd 2.3.2 FTP server [44], (2) MySQL 5.1 DB [42], and (3) Apache 2 Web server [45].

We customized the Snort configuration file to meet the AVPS requirements. All default rules that come with Snort were disabled and our own policies were added inside *local.rules*. We configured Snort to output events into a MySQL database.

The client and server are connected directly to the AVPS as shown in Figure 3. All three machines are also connected via a second network card to a switch. The controlling host is also connected to the switch to control and collect the results from all three machines.

### B. Experimental Results

In this section we show the results of experiments using the testbed described above. For each application server type, we conducted two types of experiments. The first consisted of manually submitting 10 requests to the application server. This was used to measure the average file transfer time, query response time, and throughput. The second consisted of automatically submitting 30 requests to the application server, in sequence with no think time. This process was used to measure the average, minimum, and maximum CPU utilization of the AVPS engine. All results include 95% confidence intervals.

The manual experiments considered the following four scenarios: (1) No AVPS, client and application servers are connected to a 1000-Mbps switch. (2) Client and server are connected to the AVPS but the engine is disabled, traffic is only being bridged. (3) The AVPS is enabled and no rules match the traffic (either because no policies are loaded or because the loaded policies do not trigger a violation). (4) The AVPS is enabled, detects a violation on all rules checked, and generates an alert, which is stored in a database. However, the AVPS is configured not to block the traffic. It should be noted that case (4) above is the one that generates the largest possible overhead because all rules generate a violation, an unlikely event in practice, and traffic flowing through the AVPS is not decreased due to blocking offending requests. Thus, all results presented in what follows for scenario (4) represent a worst-case performance scenario.

The automated experiments were used to measure average and maximum CPU utilization of the AVPS engine and consider the following four scenarios: (1) Same as scenario (2) above. (2) Same as scenario (3) above. (3) Same as scenario (4) above. (4) Same as scenario (4) above but the AVPS is configured to block the traffic. Case (3) above is also a worst-case performance scenario for the reasons outlined above. Case (4), the blocking case, is the ideal operational situation. In that case, blocked traffic does not contribute to network and application server load.

1) *FTP Results*: The FTP results are discussed in what follows. Table I shows the measured results for the average throughput (in KB/sec) and average transfer time (in msec) for

10 manually submitted requests using JMeter for four different file sizes: 100 KB, 1 MB, 10 MB and 100 MB.

In the case where we check against a rule (case (4) in the manual experiments), we loaded into memory the following rule that alerts when user “appserver” tries to log into a specific FTP server.

```
alert tcp any any → FTPserver any (classtype:attempted-
user; msg:“Snortinline Autonomic FTP event”;content:
“appserver”;nocase;sid:2;)
```

The elements of the rule above are (a) alert: notify the user of a violation, (b) tcp: the protocol used, (c) Any: the IP address, (d) → is the direction, (e) classtype: is the category for the type of rule, (f) msg: the description of the rule, (g) nocase: the pattern is not case sensitive, and (h) sid : unique Snort id. The syntax of Snort rules is described in [34].

From Table I, we see that the differences in the four scenarios in average throughput and average transfer time for any of the various file sizes are either statistically insignificant at the 95% confidence level (e.g., for 100 KB and 1 MB files) or are very small (e.g., less than 1.8% different for 10 MB and 100 MB files). This means that there is little or no difference between the case when the AVPS process is disabled (case (2)) and the case where the AVPS engine is enabled and all rules checked generate a violation, but traffic is not blocked (case (4)). This is expected behavior since the AVPS does not inspect packets that contain file data being transferred. It only inspects the initial administration and request commands. Thus, the AVPS has no or very little impact on throughput and transfer time.

For the CPU measurements discussed below, we used the automated submission scenario. We load into memory the following rule that blocks a user when he/she tries to access a specific FTP server using “appserver” by replacing it with “\*\*\*\*\*”.

```
alert tcp any any → FTPserver any (classtype:attempted-
user; msg:“Snortinline Autonomic FTP block”; content: “
appserver”; nocase;replace:“*****”;sid:2;)
```

Table I shows the measured average CPU utilization of the AVPS engine for 30 automated requests with zero think time using JMeter for four different file sizes: 100 KB, 1 MB, 10 MB and 100 MB.

Table I shows that the CPU utilization is negligible in most scenarios except for when the AVPS is enabled, matching, and not blocking violations for large files (i.e., 100 MB). In this case, we see an average 6.12% CPU utilization. This is considered the worst case but is still considered very small and has almost no effect on the traffic traversing or being processed. If we consider the blocking situation (the default action in an ideal AVPS deployment), we see that the CPU utilization drops to an average of 0.12%, a negligible overhead. This is expected because in this case, data packets are blocked and are not processed any further.

File Size →	100 KB	1 MB	10 MB	100 MB
<i>Average throughput (KB/Sec) with 95% confidence intervals</i>				
No AVPS, switching	327.0 ± 7.0	2811.9 ± 48.0	9834.2 ± 38.5	13395.3 ± 90.1
AVPS process not on	331.1 ± 5.3	2754.7 ± 35.6	9984.4 ± 56.3	13539.5 ± 95.0
AVPS process on but not matching	330.0 ± 6.0	2754.9 ± 45.5	9756.8 ± 29.4	13257.5 ± 57.5
AVPS matching and policy applied	332.6 ± 4.7	2746.4 ± 34.4	9841.2 ± 77.3	13300.8 ± 78.9
<i>Average transfer time (msec) with 95% confidence intervals</i>				
No AVPS, switching	307.2 ± 6.9	365.3 ± 7.2	1043.2 ± 4.2	7647.6 ± 51.6
AVPS process not on	302.8 ± 5.1	372.3 ± 4.8	1025.9 ± 6.0	7566.4 ± 52.4
AVPS process on but not matching	304 ± 5.8	372.7 ± 6.5	1049.6 ± 3.2	7725.2 ± 33.3
AVPS matching and policy applied	301.3 ± 4.4	373.4 ± 4.7	1041.1 ± 8.1	7701.2 ± 45.0
<i>Average CPU utilization (%) with 95% confidence intervals</i>				
AVPS - bridging only	0.02 ± 0.01	0.04 ± 0.03	0.05 ± 0.01	0.05 ± 0.00
AVPS enabled, not matching	0.02 ± 0.01	0.3 ± 0.06	1.44 ± 0.20	2.11 ± 0.06
AVPS enabled, matching, not blocking	0.20 ± 0.06	0.79 ± 0.17	3.90 ± 0.51	6.12 ± 0.17
AVPS enabled, matching, blocking	0.09 ± 0.02	0.12 ± 0.02	0.10 ± 0.02	0.12 ± 0.03

TABLE I: FTP results

2) *Database Server Results*: For the database server experiments we built a database of customers, orders, and order items and developed three different queries. Query Q1 returns the list of all items of all orders submitted by all customers for a total of 51,740 records. Query Q2 returns one record with the number of customers in a geographical region. This query needs to scan 50 customer records. Finally, query Q3 returns the dollar amount of all orders placed by customers in a given geographical region. While this query returns only a number, it needs to do significant work on the database to obtain the result.

Table II shows the measured average response time (in sec) for 10 manually submitted queries using mysqlslap for the three different queries and for the four scenarios described above.

For the case in which rules generate a violation alert but no traffic is blocked, we loaded into memory the following rule that alerts when a user tries to access “companyxyz” database located at a specific DB server.

```
alert tcp any any → DBserver any (classtype:attempted-
user; msg:“Snortinline Autonomic DB event”;content: “
companyxyz”;nocase;sid:2;)
```

We can see from Table II, that the worst case appears in Q1, which returns 51740 records. For Q1 the differences between no AVPS and AVPS matching is almost 5 msec, or 13% additional overhead. We consider the extra time to be small given the large number of records returned. In fact, the overhead is approximately 0.08  $\mu$ sec per record returned. For queries Q2 and Q3 we can see almost no overhead given that both only return one record. In fact, for Q3, there is no statistically significant difference at the 95% confidence level between the no AVPS and AVPS matching cases. For Q2, the difference in response time is small and equal to 1.2 msec.

It is important to note that the largest component of the response time is the transfer time over the network and not processing time at the DB server. We measured Q1, Q2, and Q3 directly at the server and we found that Q1 takes 14 msec to

Query →	Q1	Q2	Q3
<i>Average response time (msec) with 95% confidence interval</i>			
No AVPS, switching	31.6 ± 0.24	10 ± 0.31	10.6 ± 0.39
AVPS process not on	32.4 ± 0.24	10.2 ± 0.2	10.8 ± 0.57
AVPS process on but not matching	36.4 ± 0.24	11 ± 0.31	10.6 ± 0.24
AVPS matching and policy applied	36.2 ± 0.57	11.2 ± 0.2	11.2 ± 0.37
<i>Average/Maximum CPU utilization (%)</i>			
AVPS - bridging only	0.024/0.15	0.045/0.23	0.007/0.04
AVPS enabled, not matching	0.43/1.51	0.01/0.05	0.058/0.3
AVPS enabled, matching, not blocking	1.57/4.75	0.152/0.43	0.23/0.71
AVPS enabled, matching, blocking	0.220/1.49	0.262/1.14	0.221/1.05

TABLE II: DB results

execute, and Q2 and Q3 take virtually zero seconds to execute. The difference in execution time between Q1 and the other two queries lies on the fact Q1 has to output a very large number of records. Thus, the average transfer time for case (4) for query Q1 is 22 msec obtained by subtracting the average response time at the client (i.e., 36 msec) from the server execution time of 14 msec.

As before, the CPU utilization experiments use the automated submission process. In the cases where we block against a rule, we load into memory the following rule that blocks a user when he/she tries to access the “companyxyz” database located at a specific database server by replacing it with “\*\*\*\*\*”.

```
alert tcp any any → DBserver any (classtype:attempted-
user; msg:“Snortinline Autonomic DB block”; content: “
companyxyz”; nocase; eplace: “*****”;sid:2;)
```

Table II shows the measured average and maximum (after the “/”) CPU utilization of the AVPS engine for 30 automated requests with zero think time using JMeter for queries Q1, Q2, and Q3. The minimum CPU utilization was zero in all cases.

In Table II, we notice that the average CPU utilization does not fully reflect the actual CPU utilization due to the very

<b>File Size →</b>	<b>518 KB</b>
<i>Average throughput (KB/sec) with 95% confidence interval</i>	
<b>No AVPS, switching</b>	43038 ± 1675
<b>AVPS process not on</b>	33861 ± 902
<b>AVPS process on but not matching</b>	23385 ± 372
<b>AVPS matching and policy applied</b>	17938 ± 677
<i>Average transfer time (msec) with 95% confidence interval</i>	
<b>No AVPS, switching</b>	6.1 ± 0.23
<b>AVPS, process not on</b>	7.7 ± 0.21
<b>AVPS process on but not matching</b>	11.1 ± 0.18
<b>AVPS matching and policy applied</b>	14.6 ± 0.47
<i>Average CPU utilization (%) with 95% confidence interval</i>	
<b>AVPS - bridging only</b>	0.03 ± 0.04
<b>AVPS enabled, not matching</b>	0.24 ± 0.45
<b>AVPS enabled, matching, not blocking</b>	0.54 ± 1.04
<b>AVPS enabled, matching, blocking</b>	0.15 ± 0.11

TABLE III: Web results

low amount of time that it takes to process a request over the network. The maximum CPU utilization provides a better view of the actual utilization encountered. We can see again that the worst case occurs with a maximum CPU utilization of 4.75% for Q1 when the AVPS is matching but not blocking. This overhead is considered very small and almost negligible given the number of records returned. The other queries have a maximum of 1.14% utilization, which is extremely low and can almost be completely ignored. In the case of blocking (last row), we see extremely low overhead for the worst case (Q1) that has a maximum of 1.49% utilization. Again, in an ideal environment a blocking policy would be in place.

3) *Web Server Results*: The results of the experiments in a Web server environment are shown in Table III, which presents the average throughput (in KB/sec) and the average transfer time (in msec) for 10 manually submitted requests using JMeter for a Web page of 518 KB. In the cases where we check against a rule but do not block, we loaded into memory the following rule that alerts when a user tries to access the page “notallow.html” located at a specific webserver.

```
alert tcp any any → Webserver any (classtype:attempted-user;
msg:“Snortinline Autonomic web event”;
content:“notallow.html”;nocase;sid:2;)
```

Table III indicates that the average throughput is reduced by 56% when the AVPS is running, matching, and not blocking as compared with the case of no AVPS. The response time difference in that case (see Table III) increases 2.28 times. However, the increase in time units is only 8.2 msec for a large web page (i.e., 518 KB). This increase in response time is hardly noticeable by a human being. It should be noted that in the Web case, the AVPS has to inspect every single packet of a Web page.

Table III indicates that the CPU utilization results for the web case are equally low as in the previous cases.

### C. Scalability Analysis

The previous section showed experimental results obtained with our implementation of the AVPS. In this section, we use

a queuing theoretical model to examine the scalability of the AVPS under a variety of configurations not contemplated in the implementation due to resource limitations. Some examples of these configurations include many clients, many AVPS engines, and different mixes of workload. The input parameters for our queuing model, in particular the execution time and overhead of running applications protected by the AVPS, were obtained from the experiments described previously.

We assume that there are  $M$  clients that submit requests that are initially processed by one of  $N$  AVPS engines, which then send the requests to an application server (AS) (e.g., FTP server, database server, Web server). Each client pauses for an exponentially distributed time interval, called *think time*, before submitting a new request after a reply to the previous request has been received. The average think time is denoted by  $Z$ . See Figure 4 for a depiction of the model.

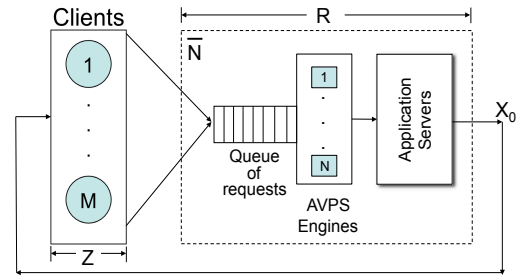


Fig. 4: AVPS analytic model.

We also assume that the average time to process a request, not counting time waiting to use resources at the AVPS and the application server, is exponentially distributed with an average equal to  $\bar{x}$ .

We can use the results of the M/M/N/M queue (see [46]) to obtain the probabilities  $p_k$  of having  $k$  requests being processed or waiting by either the AVPS or the application server. The M/M/N/M queue models a variable service rate finite-population of  $M$  request generators that alternate between two states: (1) waiting for a reply to a submitted request and (2) thinking before submitting a new request after receiving a reply to the previous request.

The probabilities  $p_k$  are then given by

$$p_k = \begin{cases} p_0 (\bar{x}/Z)^k \frac{M!}{(M-k)! k!} & 0 \leq k \leq N \\ p_0 [\bar{x}/(N Z)]^k \frac{M! N^N}{(M-k)! N!} & N < k \leq M \end{cases} \quad (1)$$

where

$$p_0 = \left[ \sum_{k=0}^N \left( \frac{\bar{x}}{Z} \right)^k \frac{M!}{(M-k)! k!} + \sum_{k=N+1}^M \left( \frac{\bar{x}}{N Z} \right)^k \frac{M! N^N}{(M-k)! N!} \right]^{-1} \quad (2)$$

We can now compute the average number,  $\bar{N}$ , of requests being processed or waiting to be processed by the AVPS + application server system as

$$\bar{N} = \sum_{k=1}^M k p_k \quad (3)$$



and the average throughput  $X_0$  as

$$X_0 = \sum_{k=1}^N \frac{k}{\bar{x}} p_k + \sum_{k=N+1}^M \frac{N}{\bar{x}} p_k. \quad (4)$$

The average response time,  $R$ , can be computed using Little's Law [31] as  $R = \bar{N}/X_0$ .

The workload intensity of such a system is given by the pair  $(M, Z)$ . An increase in the number of clients  $M$  or a decrease in the think time  $Z$  imply in an increase in the rate at which new requests are generated from the set of clients. As the processing time  $\bar{x}$  increases, contention within the system increases and requests tend to spend more time in the system instead of at the client. In the extreme case,  $p_M \approx 1$  and  $p_k \approx 0$  for  $k = 0, \dots, M-1$ . This is when saturation occurs. When that happens,  $\bar{N} \rightarrow M$ ,  $X_0 \rightarrow N/\bar{x}$ , and  $R = \bar{N}/X_0 \rightarrow M\bar{x}/N$ . In other words, the response time grows linearly with  $M$  at very high workload intensities.

In the following sections, we provide the results for three different scenarios:

- Multiple clients ( $M > 1$ ) accessing a specific application server (FTP, DB or Web) via a single AVPS ( $N = 1$ ).
- Multiple clients ( $M > 1$ ) accessing a specific application server (FTP, DB or Web) via multiple AVPS engines concurrently ( $N = 1, 2, 3, 4, 5$ ).
- Multiple clients ( $M > 1$ ) accessing a mixture of application servers (FTP, DB or Web) via multiple AVPS engines concurrently ( $N = 1, 2, 3, 4, 5$ ).

We use the  $\bar{x}$  values obtained in our measurements from Section V-B to analyze the scalability of the AVPS for an FTP server, database server and web server under the same conditions shown in the previous sections (see Table IV). Note that the values of  $\bar{x}$  used here correspond to the worst-case scenario in the automated tests, i.e., case (3) in which all rules generate a violation and an alert but traffic is not blocked.

1) *Specific Application and  $N = 1$* : Figure 5 depicts the architecture of this scenario, which discusses the performance results for the number of clients,  $M$ , varying from 1 to 30 and each client accessing a single application/element (i.e., FTP/100 MB file) via one AVPS engine.

Server type		$\bar{x}$
FTP Server	100 KB	0.360 sec
	1 MB	0.513 sec
	10 MB	1.050 sec
	100 MB	8.100 sec
DB Server	Q1	41.6 msec
	Q2	14.8 msec
	Q3	15.6 msec
Web Server	518 KB	12.1 msec

TABLE IV: Average service time  $\bar{x}$  obtained from measurements for the FTP Server, DB Server and Web Server Applications.

Figure 6 shows the average file transfer time,  $R$ , when the number of clients varies from 5 to 30 for an average think time equal to 10 sec. The AVPS is enabled, matching packets

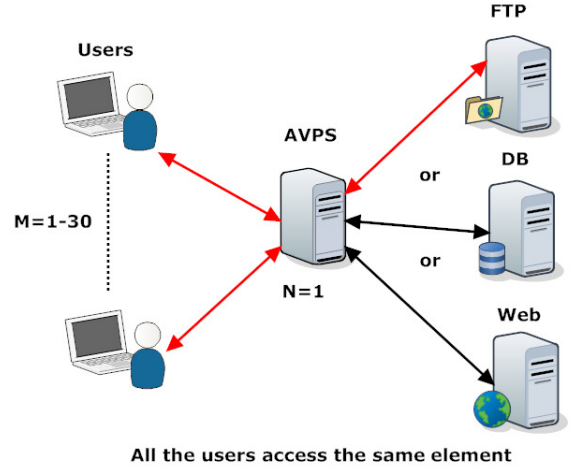


Fig. 5: Single application with one AVPS engine

against the policy, but not blocking bad transfers. If blocking were enabled, the transfer time would be reduced since some files would not be transferred. As expected, for each file size, the average transfer time increases with the file size. For large files (e.g., 100 MB) and for this value of the think time, the system is close to saturation and the average transfer time increases almost linearly with the number of clients, as discussed above. For example,  $R = 233$  sec for  $M = 30$ . This value is very close to  $30 \times \bar{x} = 30 \times 8.1 = 243$  sec. For half the number of clients,  $R$  is 111.5 sec, which is almost half the value for  $M = 30$ . But, even in this worst case, the FTP server with the AVPS system scales linearly with the number of clients.

Before saturation is reached, the increase in average transfer time is more than linear, as can be seen for example in the 10 MB file size case. For example, the value of  $R$  for  $M = 30$  is about 3.4 times higher than for  $M = 15$ . However, as  $M$  increases way past  $M = 30$  for 10-MB files, the system will saturate and the transfer time will increase linearly with  $M$ .

Figure 7 shows the average response time,  $R$ , for the result of queries Q1, Q2, and Q3 defined in Section V-B for an average think time equal to 0.1 sec. As before, the number of clients varies from 5 to 30. The number of records returned by queries Q1-Q3 are 51740, 1, and 1, respectively. Q3 is a much more complex query and requires more database processing time. Thus, its average response time is slightly higher than that for Q2, even though both queries return the same amount of data. The graph indicates that for 30 clients and for Q1, the system is very close to saturation and the average transfer time is very close to be proportional to  $M$ . In fact,  $R = 1.148$  sec  $\approx 30 \times \bar{x} = 30 \times 0.0416 = 1.248$  sec. Queries Q2 and Q3 do not return enough records to push the system to saturation and therefore we see a more than linear increase in transfer time as a function of  $M$  for the values shown in the graph.

Figure 8 shows the average transfer time  $R$  for a 518-KB Web page and for an average think time equal to 1 sec. As before, the number of clients varies from 5 to 30. The

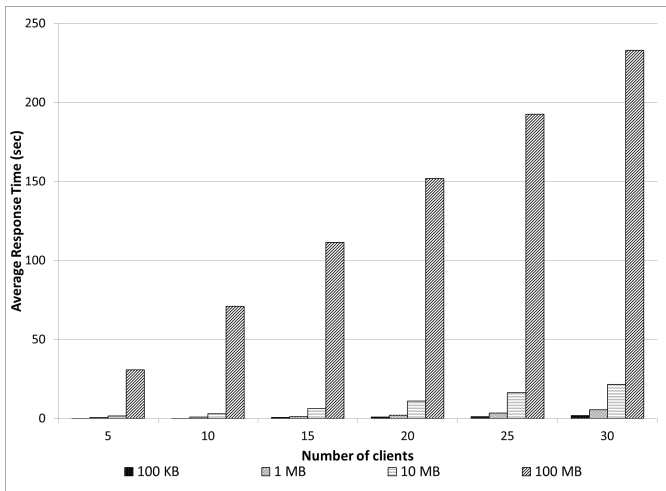


Fig. 6: Average file transfer time vs. number of clients for various file sizes. The average think time is 10 sec.

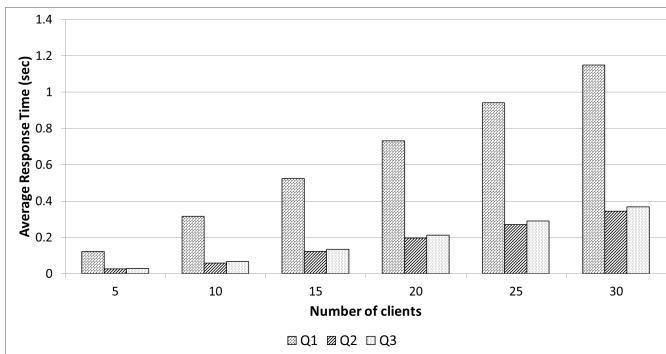


Fig. 7: Average database query result transfer time vs. number of clients for three different queries. The average think time is 0.1 sec.

graph indicates that the increase in transfer time is negligible between 5 and 30 clients. While  $R$  increases linearly with the number of clients, the rate of increase is mainly due to increased congestion at the Web server and not to AVPS overhead, which is small (8.2 msec) and hardly noticeable by a human being.

2) *Specific Application and  $N=1-5$* : Figure 9 depicts the architecture of this scenario. The performance results discussed here are for 1 to 30 clients accessing a single application/element (e.g., FTP/100 MB file) via 1 to 5 AVPS engines.

Figure 10 shows the average file transfer time when the number of clients,  $M$ , varies from 5 to 30 for an average think time equal to 10 sec, for a 100-MB file transfer, and for a number of AVPS engines,  $N$ , varying between 1 and 5. The AVPS is enabled, matching packets against the policy, but not blocking bad transfers. If blocking were enabled the transfer time would be reduced since some files would not be transferred. As expected, the average transfer time decreases substantially, and in a non-linear way, with the increase in the number of AVPS engines, especially for a higher number of

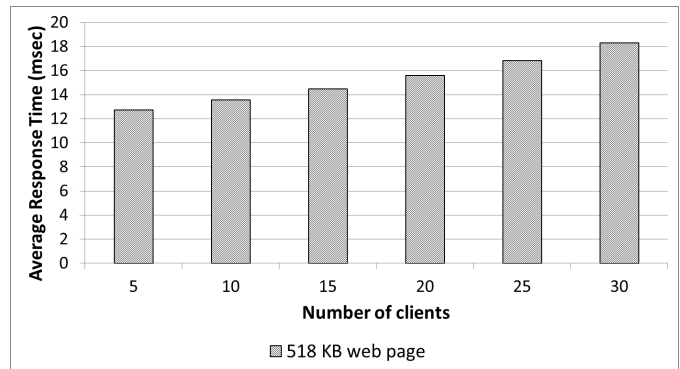


Fig. 8: Average web transfer time vs. number of clients for a 518-KB Web page. The average think time is 1 sec.

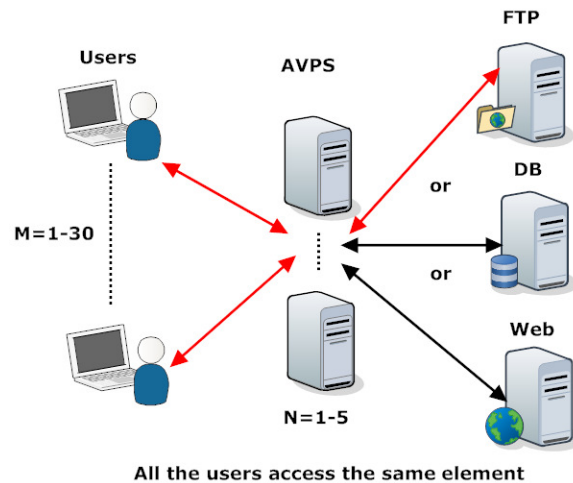


Fig. 9: Single application mode architecture with multiple AVPS engines

clients. This is due to the fact that more clients generate more contention at the AVPS. The addition of more AVPS engines reduces contention. For example, for  $M = 30$  the response time decreases by 83% as one goes from one to five AVPS engines. For any value of the number of clients, there is a value  $N^*$  of the number of AVPS engines that does not produce any significant reduction in response time because contention has already been eliminated. At that point, the response time must be equal to the service time  $\bar{x}$ . For the case shown in Figure 10, this value is  $\bar{x} = 8.1$  sec (see Table IV for the average service time for 100-MB files). For example, for  $M = 5$ ,  $N^* = 3$  and for  $M = 10$ ,  $N^* = 5$ .

The curves of Figure 10 can also be used to determine the adequate number of AVPS engines for a desired average response time. For example, for 25 clients, 2 AVPS engines would be required for the average response time not to exceed 100 sec.

Figure 11 shows the average response time for queries of type Q1 defined in Section V-B for an average think time equal to 0.1 sec and the number of AVPS engines  $N$  varying between 1 and 5. As before, the number of clients varies from 1 to 30.

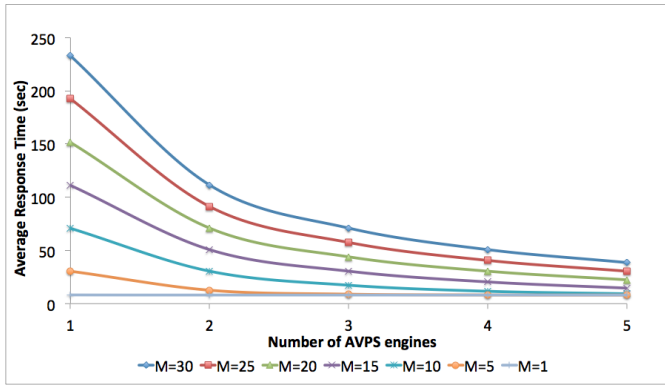


Fig. 10: Average file transfer time vs. number of AVPS engines for a 100-MB file. The number of clients varies from 1 to 30. The average think time is 10 sec.

The results are very similar to those of the FTP case. All curves must eventually converge to 41.6 msec (see Table IV for the average service time for queries of type Q1) when  $N = N^*$ . For example,  $N^* = 3$  for  $M = 5$  and  $N^* = 4$  for  $M = 10$ . The average response exhibits an 87% reduction for 30 clients as the number of AVPS engines increases from 1 to 5.

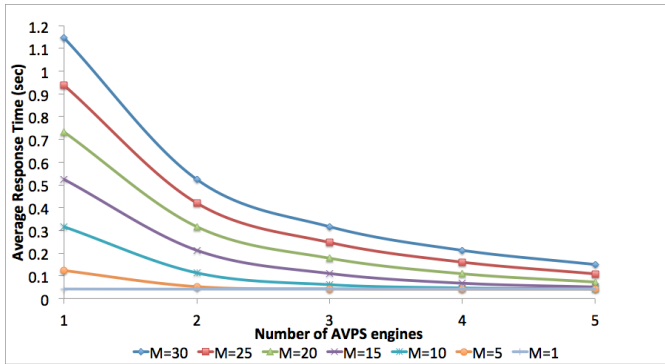


Fig. 11: Average query Q1 response time vs. number of AVPS engines for various values of the number of clients. The average think time is 0.1 sec.

Figure 12 shows the average transfer time,  $R$ , for a 518-KB Web page, for an average think time equal to 1 sec, and for the number of AVPS engines  $N$  varying between 1 and 5. As before, the number of clients varies from 1 to 30. The graph indicates that all curves (1-30 clients) almost converge to the value of 12.1 msec (see Table IV for the average service time for a 518-KB Web page transfer) when a second AVPS engine is added into the system. Thus,  $N^* = 2$  for all values of the number of clients in this case. For 30 clients, the reduction in response time is about 33% as an additional AVPS engine is added.

3) *Mixed Application and  $N = 1, \dots, 5$* : Figure 13 depicts a scenario in which users access any of the three applications.

We discuss here the performance results of a scenario in which 1 to 30 clients access multiple applications (e.g., FTP,

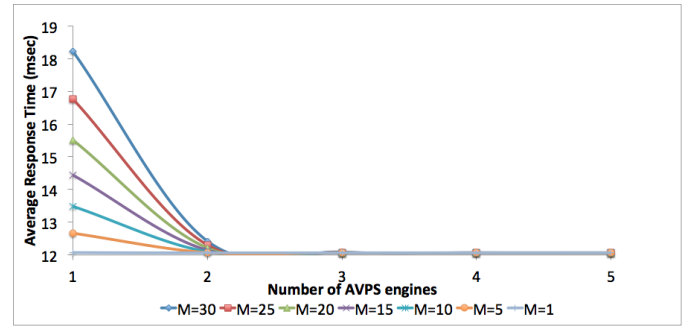


Fig. 12: Average Web page transfer time vs. number of AVPS engines for various values of the number of clients for a 518-KB Web page. The average think time is 1 sec.

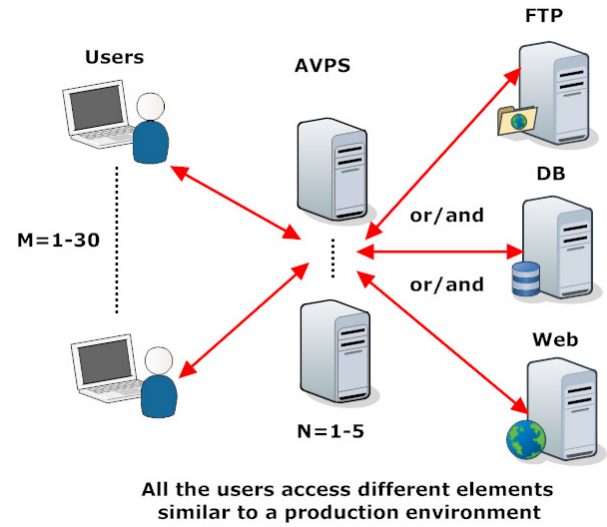


Fig. 13: Mixed application scenario with multiple AVPS engines.

DB, and Web) with multiple files sizes and query types for a number of AVPS engines varying from 1 to 5.

Figures 14, 15, and 16 show, respectively, the results of three different experiments:

- The average file transfer time when the number of clients varies from 1 to 30 for an average think time equal to 10 sec, the file transfer is for a mix of 100 KB, 1MB, 10 MB and 100 MB files, and the number of AVPS engines  $N$  varies between 1 and 5.
- The average query response time for a mix of Q1, Q2 and Q3 queries when the number of clients varies from 1 to 30 for an average think time equal to 0.1 sec, and for a number of AVPS engines  $N$  varying between 1 and 5.
- The average transfer time for a mix of FTP downloads of files of size 100 KB, 1MB, 10 MB, 100 MB, queries of type Q1, Q2, Q3 and a 518-KB Web page. The number of clients varies from 1 to 30 for an average think time equal to 5.16 sec and the number of AVPS engines  $N$  varying between 1 and 5.

In all three cases, the AVPS is enabled, matching packets



against the policy, but not blocking bad transfers. If blocking were enabled the transfer time would be reduced since some files would not be transferred.

Similar to previous results, the average transfer time decreases substantially with the increase in  $N$ . We notice, as expected, that when the number of clients increases the performance gain increases when additional AVPS engines are used. As before, there is a point after which additional AVPS engines do not improve performance.

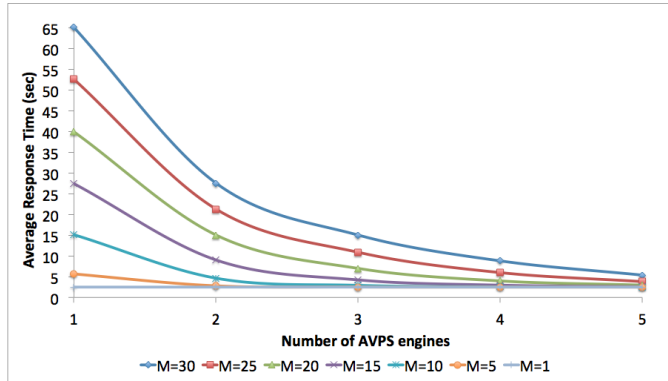


Fig. 14: Average FTP transfer time vs. number of AVPS engines for a mix of 100 KB, 1 MB, 10 MB, and 100 MB file downloads. The number of clients varies from 1 to 30. The average think time is 10 sec.

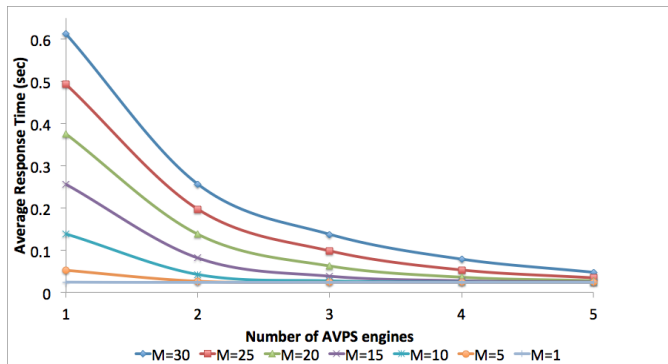


Fig. 15: Average query response time vs. number of AVPS engines for a mix of Q1, Q2 and Q3. The number of clients varies from 1 to 30. The average think time is 0.1 sec.

## VI. CONCLUSION AND FUTURE WORK

This paper presented a scalable AVPS framework to defeat the insider threat. The AVPS is an inline mechanism that inspects traffic between insider clients and servers. The AVPS uses low level rules in the form of ECAs, implemented as Snort rules in our prototype. An offline process uses supervised learning to learn high-level rules that are automatically converted into one or more low level rules.

The paper also presented a performance evaluation assessment for three different application servers. The performance

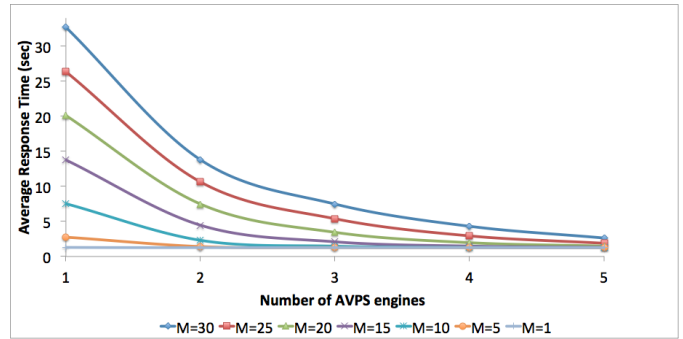


Fig. 16: Average transfer time vs. number of AVPS engines for a mix of different applications (FTP, DB, and Web requests). The number of clients varies from 1 to 30. The average think time is 5.16 sec.

assessment measured average transfer times, average throughput, and CPU utilization as well as 95% confidence intervals for all three measurements.

The experiments showed that: (1) The impact on the average transfer time and throughput for FTP transfers is either negligible at the 95% confidence level or very small (i.e., less than 1.8%). (2) The response time impact on database queries is heavily dependent on the number of records returned by the queries. For queries that return a very large number of records (e.g., over 51,000), the response time increase is 13% on average. However, this amounts to only 0.08  $\mu$ sec on average per record returned. (3) When a Web server is accessed through the AVPS system, the response time for a large Web page (e.g., 518 Kbytes) increases by 8.2 msec, an amount hardly noticeable by a human being. (4) The average and maximum CPU utilization of the AVPS engine are very small in all cases tested, not exceeding 7%.

We also presented an M/M/N/M queuing analytical scalability model for three different cases with a varying number of applications, users, and AVPS engines and generated average response times curves for all different cases. The scalability and performance model showed that the AVPS framework can easily scale horizontally to achieve the desired performance level. The model also showed that for each number of clients, there is an optimal number of AVPS engines that totally eliminates congestion and minimizes response time. Using more than that number of AVPS engines does not improve performance any further.

Our results also showed that there is very low overhead incurred when the AVPS is in-line between the clients and the application servers. We used worst-case scenarios in our analysis by considering situations in which all checked rules trigger a violation and generate an alert, but do not block incoming traffic. Blocking traffic in violation situations, which is the normal operational approach, reduces the load on the network and on the AVPS engine and improves performance.

We are currently looking at model based architectures, typically used in self-optimizing systems, and the effects of rule complexity on the overall performance of the system.

## REFERENCES

- [1] F. Sibai and D. Menascé, "A scalable architecture for countering network-centric insider threats," in *SECURWARE 2011, The Fifth Intl. Conf. Emerging Security Information, Systems and Technologies*, Nice/Saint Laurent du Var, France, 2011, pp. 83–90.
- [2] "zdnet," 2010, last accessed on 6/17/2012. [Online]. Available: <http://www.zdnet.com/blog/perlow/wikileaks-how-our-government-it-failed-us/14988>
- [3] F. Sibai and D. Menascé, "Defeating the insider threat via autonomic network capabilities," in *Communication Systems and Networks (COM-SNETS), 2011 Third Intl. Conf.* Bangalore, India: IEEE, 2011, pp. 1–10.
- [4] M. Huebscher and J. McCann, "A survey of autonomic computing - degrees, models, and applications," *ACM Comp. Surveys*, Vol. 40, Issue 3, pp. 1–28, 2008.
- [5] G. Jabbour and D. Menascé, "Policy-Based Enforcement of Database Security Configuration through Autonomic Capabilities," in *Proc. Fourth Intl. Conf. Autonomic and Autonomous Systems*. IEEE Computer Society, 2008, pp. 188–197.
- [6] —, "The Insider Threat Security Architecture: A Framework for an Integrated, Inseparable, and Uninterrupted Self-Protection Mechanism," in *Proc. 2009 Intl. Conf. Computational Science and Engineering-Volume 03*. Vancouver, Canada: IEEE Computer Society, 2009, pp. 244–251.
- [7] M. Engel and B. Freisleben, "Supporting autonomic computing functionality via dynamic operating system kernel aspects," in *Proc. 4th Intl. Conf. Aspect-oriented Software Development*. Chicago, IL, USA: ACM, 2005, p. 62.
- [8] Y. Al-Nashif, A. Kumar, S. Hariri, G. Qu, Y. Luo, and F. Szidarovsky, "Multi-Level Intrusion Detection System (ML-IDS)," in *Intl. Conf. Autonomic Computing*, 2008. Karlsruhe, Germany: IEEE, 2008, pp. 131–140.
- [9] R. He, M. Lacoste, and J. Leneutre, "A Policy Management Framework for Self-Protection of Pervasive Systems," in *2010 Sixth Intl. Conf. Autonomic and Autonomous Systems*. Cancun, Mexico: IEEE, 2010, pp. 104–109.
- [10] V. Paxson, R. Sommer, and N. Weaver, "An architecture for exploiting multi-core processors to parallelize network intrusion prevention," in *Sarnoff Symposium, 2007 IEEE*. Princeton, NJ: IEEE, 2007, pp. 1–7.
- [11] K. Scarfone and K. Mell, "Guide to intrusion detection and prevention systems (idps)," *NIST Special Publication*, vol. 800, no. 2007, p. 94, 2007.
- [12] K. Xinidis, K. Anagnostakis, and E. Markatos, "Design and implementation of a high-performance network intrusion prevention system," *Security and privacy in the age of ubiquitous computing*, pp. 359–374, 2005.
- [13] G. Vasiliadis, S. Antonatos, M. Polychronakis, E. Markatos, and S. Ioannidis, "Gnort: High performance network intrusion detection using graphics processors," in *Recent Advances in Intrusion Detection*. Boston, MA, USA: Springer, 2008, pp. 116–134.
- [14] S. Shaikh, H. Chivers, P. Nobles, J. Clark, and H. Chen, "Towards scalable intrusion detection," *Network Security*, vol. 2009, no. 6, pp. 12–16, 2009.
- [15] X. Chen, Y. Wu, L. Xu, Y. Xue, and J. Li, "Para-snort: A multi-thread snort on multi-core ia platform," in *Parallel and Distributed Computing and Systems*. ACTA Press, 2009.
- [16] G. Jedhe, A. Ramamoorthy, and K. Varghese, "A scalable high throughput firewall in fpga," in *Field-Programmable Custom Computing Machines, 2008. FCCM'08. 16th International Symposium on*. Palo Alto, California, USA: Ieee, 2008, pp. 43–52.
- [17] H. Gobjuka and K. Ahmat, "Fast and scalable method for resolving anomalies in firewall policies," in *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*. Shanghai, China: IEEE, 2011, pp. 828–833.
- [18] B. Wun, P. Crowley, and A. Raghunth, "Parallelization of snort on a multi-core platform," in *Proceedings of the 5th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*. Princeton, NJ, USA: ACM, 2009, pp. 173–174.
- [19] M. Alam, Q. Javed, M. Akbar, M. Rehman, and M. Anwer, "Adaptive load balancing architecture for snort," in *Networking and Communication Conference, 2004. INCC 2004. International*. Lahore, Pakistan: IEEE, 2004, pp. 48–52.
- [20] D. Day and B. Burns, "A performance analysis of snort and suricata network intrusion detection and prevention engines," in *ICDS 2011, The Fifth Intl. Conf. Digital Society*. Gosier, Guadeloupe, France: IARIA, 2011, pp. 187–192.
- [21] K. Salah and A. Kahtani, "Performance evaluation comparison of snort nids under linux and windows server," *Journal of Network and Computer Applications*, vol. 33, no. 1, pp. 6–15, 2010.
- [22] D. Menascé, "Security performance," *IEEE Internet Computing*, vol. 7, no. 3, pp. 84–87, 2003.
- [23] "viewSSLd," 2012, last accessed on 6/17/2012. [Online]. Available: <http://sourceforge.net/projects/viewssld/>
- [24] "Netintercept, Niksun Inc." 2012, last accessed on 6/17/2012. [Online]. Available: <http://www.niksun.com/product.php?id=16>
- [25] "Ettercap, Sourceforge," 2009, last accessed on 6/17/2012. [Online]. Available: <http://ettercap.sourceforge.net/index.php>
- [26] F. Sibai and D. Menascé, "Countering network-centric insider threats through self-protective autonomic rule generation," in *IEEE Sixth Intl. Conf. Software Security and Reliability (SERE 2012)*. IEEE, 2012, p. 10.
- [27] F. Sibai, "Defeating insider attacks via autonomic self-protective networks," Ph.D. dissertation, George Mason University, Fairfax, VA, 2012.
- [28] "Network Critical V-Line TAP, Network Critical Solutions Limited," 2012, last accessed on 6/17/2012. [Online]. Available: <http://www.networkcritical.com/Products/Bypass.aspx>
- [29] "IBM Proventia Network Intrusion Prevention System , IBM," 2011, last accessed on 6/17/2012. [Online]. Available: <http://www-01.ibm.com/software/tivoli/products/network-multifunction-security/>
- [30] "Cisco ASA, Cisco Systems," 2011, last accessed on 6/17/2012. [Online]. Available: <http://www.cisco.com/en/US/products/ps6120/index.html>
- [31] L. Kleinrock, *Queueing systems, volume 1: theory*. John Wiley & Sons, 1975.
- [32] "Snort performance, Sourcefire, Inc ,," 2010, last accessed on 6/17/2012. [Online]. Available: <http://www.snort.org/assets/168/LW-hakin9-custm-rules-2010.pdf>
- [33] "Snort tuning, Sourcefire, Inc ,," 2010, last accessed on 6/17/2012. [Online]. Available: [http://www.snort.org/assets/127/Snort\\_Perf\\_Tuning\\_webinar\\_Final.pdf](http://www.snort.org/assets/127/Snort_Perf_Tuning_webinar_Final.pdf)
- [34] "Snort manual, Sourcefire, Inc ,," 2009, last accessed on 6/17/2012. [Online]. Available: [http://www.snort.org/assets/120/snort\\_manual.pdf](http://www.snort.org/assets/120/snort_manual.pdf)
- [35] "Barnyard," 2009, last accessed on 6/17/2012. [Online]. Available: <http://barnyard.sourceforge.net/>
- [36] "JMeter," 2012, last accessed on 6/17/2012. [Online]. Available: <http://jakarta.apache.org/jmeter/>
- [37] "MySQL Slap, Oracle Corporation," 2012, last accessed on 6/17/2012. [Online]. Available: <http://dev.mysql.com/doc/refman/5.1/en/mysqlslap.html>
- [38] "Snort, Sourcefire, Inc ,," 2010, last accessed on 6/17/2012. [Online]. Available: <http://www.snort.org/snort>
- [39] "Bro Intrusion Detection System, Lawrence Berkeley National Laboratory," 2011, last accessed on 6/17/2012. [Online]. Available: <http://www.bro-ids.org/>
- [40] "Event Monitoring Enabling Responses to Anomalous Live Disturbances (EMERALD), SRI Intl." 2012, last accessed on 6/17/2012. [Online]. Available: <http://www.csl.sri.com/projects/emerald/>
- [41] "Iptables, netfilter," 2010, last accessed on 6/17/2012. [Online]. Available: <http://www.netfilter.org/>
- [42] "MySQL DB, Oracle Corporation," 2012, last accessed on 6/17/2012. [Online]. Available: <http://www.mysql.com/>
- [43] "BASE Project, Basic Analysis and Security Engine," 2008, last accessed on 6/17/2012. [Online]. Available: <http://base.secureideas.net/>
- [44] "Vsftpd," 2012, last accessed on 6/17/2012. [Online]. Available: <http://vsftpd.beasts.org/>
- [45] "Apache 2, The Apache Software Foundation," 2012, last accessed on 6/17/2012. [Online]. Available: <http://httpd.apache.org/>
- [46] D. Menascé and V. Almeida, *Capacity Planning for Web Services*. Prentice Hall, 2002.

## Advances in Protecting Remote Component Authentication

Rainer Falk, Steffen Fries  
Corporate Technology  
Siemens AG  
D-81739 Munich, Germany  
e-mail: {rainer.falk|steffen.fries}@siemens.com

**Abstract**—Component authentication allows verifying the originality of various components being part of a machine or a system, or being connected to control equipment. Various technologies are available, ranging from holograms, hidden marks, special inks to cryptography-based component authentication. Typical applied cryptography-based mechanisms employ a challenge-response-based component authentication mechanism. These component authentication mechanisms have been designed originally for local genuineness verification, i.e., for an authentication performed in direct vicinity of the component to be verified. However, it may be useful to support also a remote component authentication, e.g., to verify the integrity of the control system including its periphery from a central monitoring station. This paper describes an attack on a challenge-response component authentication protocol when using it in addition for a remote component authentication. A new security measure, that binds a challenge value to a specific remote verifier, is described to prevent this attack. The challenge value for which the response is calculated by the component authentication mechanism can therefore not be selected by the remote verifier. This has the advantage on one hand that a potential remote adversary cannot use the component as oracle to collect challenge response pairs. On the other hand, the response value can be provided to the verifier directly.

**Keywords**—device authentication, counterfeiting, tunneled authentication

### I. INTRODUCTION

Authentication is an elementary security service proving that an entity in fact possesses a claimed identity. Often natural persons are authenticated. The basic approaches a person can use to prove a claimed identity are by something the person knows (e.g., a password), by showing something the person has (e.g., passport, authentication token, smart card), or by exposing a physical property the person has (biometric property, e.g., a fingerprint, voice, iris, or behavior). Considering the threat of counterfeited products (e.g., consumables, replacement parts) and the increasing importance of ubiquitous machine-oriented communication, also physical objects need to be authenticated in a secure way. Various technologies are used to verify the authenticity of products, e.g., applying visible and hidden markers, using security labels (using e.g., security ink or holograms), and by integrating cryptographic authentication functionality (wired product authentication token or RFID (Radio Frequency Identification) authentication tag)).

One important driver for verifying the authenticity of products is safety. For example, counterfeited electrical components as electrical switches or fuses can cause physical damage when they do not fulfill electrical safety norms (e.g., by causing electric shock to humans or a fire). Other examples are provided through electric safety devices as e.g., overvoltage protecting units or an earth leakage circuit breaker that do not provide reliably the expected functionality. Further examples can also be provided through metering or measurement systems like Phasor-Measurement-Units, which measure voltage and current magnitude and phase angle values at diverse locations in the energy transmission grid. If they report wrong values, the reliability of transmission grids may be endangered. Component authentication may be used also by vendors to identify replacement parts or consumables like ink or toner.

An approach for remote component authentication has already been described in [1] using a challenge-response authentication of a physical object. Focus of this paper is a further elaboration of this approach to provide more background to and insight into the proposed solution for.. Authentication of a physical object has the advantage that control or supervisory equipment can automatically verify the authenticity of installed components. Local object authentication is used widely e.g., for authenticating battery packs or printer consumables (toner / ink cartridges). Here, cryptographic challenge-response based component authentication is applied. Highly cost-optimized solutions are commercially available that allow to use these technologies also in low-cost devices. However, local object authentication are, often not being designed to protect against man-in-the-middle attacks as these may not be seen as relevant in the targeted use case. Hence, the applied protocols or methods may not directly be applicable to remote authentication. Here, especially Man-in-the-Middle attacks are in scope, if public networks are traversed.

Section II gives an overview on challenge-response based component authentication. The usage scenario for remote component authentication is described in Section III. This section also considers typical available technical solutions, and their susceptibility to man-in-the-middle attacks when used for remote component authentication by different verifiers. A new, simple to implement countermeasure protecting against man-in-the-middle attacks is described in Section IV. It enables the re-using of highly cost-optimized component authentication also for remote component authentication, i.e., for a usage scenario for which it has not

been designed in the first place. This enables to use extremely simple and therefore cost-efficient hardware-based device security mechanisms for new usages that have not been addressed originally. The countermeasure can be applied in particular even in those cases when the verifier needs access to the unmodified, raw response value. The application to IP-based smart objects is described in Section VI, providing a highly optimized basis for a secure device identity within the Internet of Things. Related work is summarized in Section VII, before giving a summary and outlook in Section VIII.

## II. COMPONENT AUTHENTICATION

This subsection provides an overview about component authentication in general and specifically about commercially available implementation examples. Besides the pure mechanism overview application, some use cases are shortly presented to provide more insight into the value component authentication can provide.

### A. Overview

Components of a machine (internal or attached) shall be identified securely. This requirement is known for components like ink cartridges, batteries. In industrial machines it applies to replacement parts, sensors, actor devices. Authentication of a device allows a reliable identification of original products.

For authentication a challenge value is sent to the object to be authenticated. This object calculates a corresponding response value which is returned to the requestor and verified. The response can be calculated using a cryptographic authentication mechanism or by using a physical unclonable function (PUF).

For cryptographic authentication different mechanisms may be used. Examples are keyed hash functions like HMAC-SHA1 or symmetric ciphers in cipher block chaining (CBC-MAC, see [10]) mode or symmetric ciphers in Galois counter mode (GMAC, see [19]) up to digital signatures. For the symmetric ciphers AES would be a suitable candidate. Common to keyed hashes or symmetric key based cryptographic authentication approaches is the existence of a specific key, which is only available to the object to be authenticated and the verifier. One resulting requirement from this fact is obviously the protection of the applied key, as the leakage of this key leads to attack options, which can be easily exploited. This may in turn lead to a higher administrative effort on both sides, the component and the verifier. Another requirement that may be derived is that the key should ideally be unique for a dedicated component and not only for a batch or version of that component. A reasoning for this requirements is that if the key of one component gets compromised, an attack is only successful for this specific component, but not for other components from the same series. Also asymmetric cryptography can be used for component authentication. A suitable procedure based on elliptic curves has been described in [17].

As only an original product can determine the correct response value corresponding to a challenge, the product

entity or a dedicated part of the product is thereby authenticated.

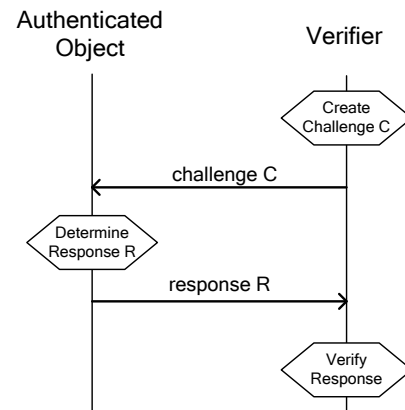


Figure 1. Challenge-Response Object Authentication

Figure 1 shows the schematic data flow between a verifier and an object to be authenticated. The verifier sends a random challenge to the component that determines and sends back the corresponding response. The verifier checks the response. Depending on the result, the component is accepted as genuine or it is rejected. The further handling upon detecting that a component or system has been compromised is a matter of the security policy and is not detailed here. Nevertheless, the reaction on detecting a failed authentication is use case specific and may vary from an immediate rejection of the component (like in the case of faked batteries, to ensure the safety of the device using the battery) to a graceful degradation mode of operation (e.g., in case of toner cartridges certain modes may not work, but the general functionality is still provided).

### B. Implementation Examples

As stated in the previous section various cryptographic mechanisms can be used to realize a challenge-response product authentication. Basically, symmetric cryptography, asymmetric cryptography, or physical unclonable functions can be used. While in the case of symmetric cryptography also the verifier is in possession of the cryptographic device key and can therefore calculate the expected response value, in case of asymmetric cryptography or PUFs the verifier might not be in a position to calculate the correct response himself. He has only the option to verify the received response.

This has consequences on whether it is possible to include binding parameters in the response, i.e. to calculate a derived response value. In the symmetric case, the verifier could calculate the expected response and perform the expected response modifications and compare this obtained expected result with the actually received result. However, in the PUF and asymmetric cases, this is not possible as the verifier can perform only a verification operation on the received result, but cannot determine a valid response on its own. The verifier needs therefore access to the original, unmodified response value to perform the verification

operation. It is not possible to use a derived response value, e.g., by using a keyed hash function or a key derivation function that uses freely definable binding parameters during the response derivation, without letting the verifier know.

Implementation examples of product authentication are summarized in the following list, providing one example per category (symmetric, asymmetric, and PUF based authentication):

- Atmel CryptoAuthentication [1], [3]: A symmetric-key based authentication is performed, intended for example for authenticating battery packs. A challenge-response authentication based on the SHA256 hash algorithm is implemented to compute a keyed digest for the provided challenge value. The input parameter to the SHA256 algorithm is the concatenation of the secret key, the challenge value, and optionally other chip-specific data (serial number, fuses). The challenge is an arbitrary 256 bit value selected by the verifier.
- Infineon ORIGA [4]: An asymmetric authentication based on elliptic curves is performed. A cryptographic operation is performed by the product to be authenticated using the product's private key. The verifier checks the received response using the corresponding public key by performing a cryptographic verification operation on the received value. The verifier does not need access to the products private key. The product itself may provide its public key as a digital certificate (using internal memory), allowing for an offline verification of the response.
- Verayo RFID Tag "Vera M4H" [5]: An integrated circuit comprising a physically unclonable function is used to determine a response value depending on the challenge value and hardly to reproduce physical characteristics of the product to be authenticated. Therefore, no cryptographic key has to be stored on the RFID tag as a physical fingerprint of the RFID tag is employed.

### C. Applications / Use Cases

Applications of component authentication have already been given in the introduction targeting safety on one hand and protection of business models (and interests) on the other. As shown, a reliable identification of products is needed in various use cases. For safety reasons, components can be verified to ensure that no counterfeited products or product components have been installed. Detection of unverifiable product components may not necessarily lead to outages. Depending on the use case, the component may be operated with safe, conservative operating conditions (e.g., maximum charging current of battery pack) to prevent damage.

Component authentication can also be leveraged for centralized control, as the information about originally or falsified system parts may be used to provide system integrity information as part of an inventory management. Moreover, the component authentication can also be used to support the authenticated setup of a protected communication session for

field level device communication. This approach is outlined in the following section.

## III. REMOTE COMPONENT AUTHENTICATION

One important class of use cases is remote component authentication. Here, a machine equipped with or connected to several field devices (sensors, actuators) performs local component authentication of installed machine parts or connected periphery in the first place. Additionally, it also supports a remote component authentication by a supervisory system, e.g., a control center or an inventory management. Remote component authentication is required in scenarios where sensors are used in a widely dispersed area. One example for such sensors are quality monitoring devices like synchrophasors (Phasor Measurement Units – PMU), which are used in the power transmission network to measure the phase angle and magnitude of sinus waves of voltage and currents. Based on this information, the network (and system) condition may be deduced.

### A. Use Case Description

In a remote object authentication, the verifier is distant to the object to be authenticated. The challenge and response values are encoded in messages that are transported over a communication network, e.g., an IP-based network, see Figure 2.

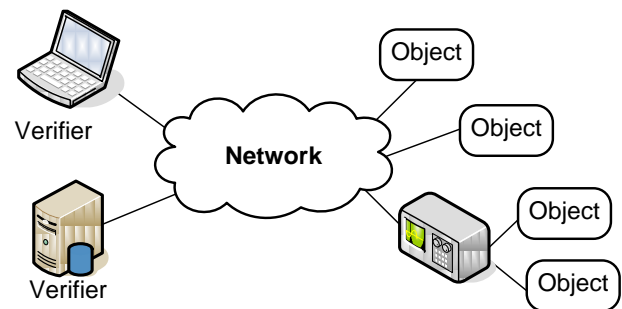


Figure 2. Remote Object Authentication

A verifier may be a service technician performing remote maintenance, or an automatic device tracking server or an inventory management server, see left part of Figure 2. The objects to be authenticated may be connected directly to the network, or they may be attached to an intermediary device, e.g. a programmable logic controller, see right part of Figure 2.

The challenge response component authentication operation, i.e., the calculation of the response value for a given challenge value, is performed by the object to be authenticated as described above. However, here the verifier is not in close vicinity to the object to be authenticated so that an intermediary (man in the middle) may intercept and manipulate the exchange of challenge and response values. In some cases, a protected communication channel may be used between the verifier and the intermediary, e.g., IPsec or SSL/TLS (Secure Socket Layer / Transport Layer Security), to transport challenge and response values over an encrypted communication channel.



### B. Man-in-the-Middle Susceptibility

In the case of remote object authentication, an adversary node may manipulate the exchange of challenge and response values when they are sent over a communication network. The adversary may act as remote verifier towards an object to be authenticated, and as authenticated object towards a well-behaving remote verifier. Such a malicious remote verifier acts as a man-in-the-middle attacker. Already the simple forwarding of challenge and response messages may constitute an attack, see Figure 3.

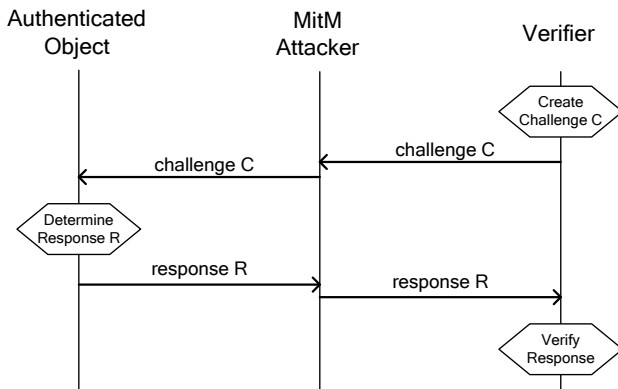


Figure 3. MitM Attack on Object Authentication

An adversary node as “man-in-the-middle” forwards unchanged messages towards and from the genuine object that is authenticated. The remote verifier having authenticated the object as genuine assumes that it is communicating in fact with the device in the following data exchange. However, in fact it is communicating with the adversary. An attacker can use an arbitrary, remote genuine object as an oracle that provides valid responses for an arbitrary challenge value. In consequence, any remote entity that has access to the object authentication functionality of a genuine object can act as a man-in-the-middle and may authenticate itself as the authenticated object if it has a sufficient number of challenge and response pairs. Hence, the remote object authentication can be manipulated.

When furthermore the authentication response is used for deriving cryptographic session keys, these keys could be derived by an attacker as well.

The fact that such a simple challenge-response authentication is prone to man-in-the-middle attacks is well known and documented also in the corresponding product documentation. For example, the man-in-the-middle attack is mentioned in [6]. In the considered usage environment where authenticated object and verifier are in direct physical vicinity, the attacker needs both a direct physical access to the attacked object and measurement equipment like e.g., a logic analyzer to analyze the information exchanged between the components. This increases the overall effort of the attack. The attack becomes relevant when the component authentication mechanism is applied within a usage environment in which not only a single verifier exists, but

where a component may be authenticated by multiple verifiers, at least one of them connected only remotely or at least not in close vicinity of the object.

### IV. EXAMPLE FOR CRYPTOGRAPHIC BINDING REQUIREMENTS

Binding cryptographic data as e.g., a session key to a specific usage context is a basic countermeasure to prevent attacks in which valid cryptographic data is applied by an adversary in a different usage context. The general need for cryptographic binding is motivated by describing a well-known weakness of the TLS protocol. Transport Layer Security (TLS) is a very popular security protocol, which is used to protect web transactions in applications like online banking, to protect the mail communication via IMAP (Internet Message Access protocol), to realize VPNs (Virtual Private Networks) or for remote administration. Meanwhile the protocol is available as standard in version 1.2 as RFC 5246 [7].

Early November 2009, a vulnerability has been discovered, allowing an attacker to inject data into a TLS connection without being noticed by the client. Such attacks were facilitated by a protocol weakness concerning renegotiation of security parameters. Renegotiation is a TLS feature to establish fresh security parameters for an existing TLS session. The problem arose due to the missing cryptographic binding between the initially negotiated security parameters and the new parameter set resulting from the renegotiation process. This can be exploited by an attacker in a man-in-the-middle attack. A possible attack scenario – a request to a web server – is explained in the following, see Figure 4.

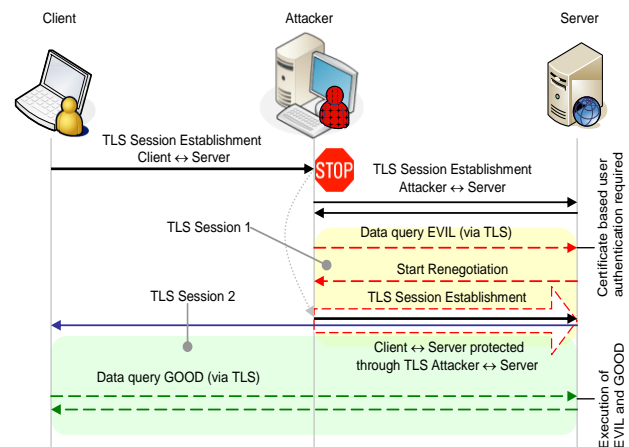


Figure 4. Man in the Middle (MitM) Attack on TLS Protocol

A potential attacker controlling the data path between a Web client (Web browser) and a Web server is waiting for a connection attempt by a client. As soon as the client establishes a TLS connection to the server, the attacker delays the client request. In a variety of applications this connection is used with unilateral authentication, i.e., only the server authenticates as part of the TLS handshake. Now, the attacker himself establishes an own TLS connection to

the server, which is (as the original connection attempt by the client) unilaterally authenticated by the server side. So the client itself has not been authenticated by the server. As soon as the TLS connection has been established, the attacker sends a request *EVIL* over the established TLS connection. This request requires the authentication of the client. If the client is to be authenticated using certificates, it can be directly done within the TLS handshake, requiring the certificate based authentication of the client by the server. The attacker, however, does not possess the required client credentials. This client authentication is now invoked by the server through starting the TLS renegotiation. The *EVIL* request, however, which has not been authenticated at this point in time, is stored by the server and will be executed only after a successful client authentication. Web servers are typically configured in this way, i.e., to request a client authentication not at the beginning of a session, but only when a client performs a restricted operation, e.g., when data from an access protected directory shall be read.

In the attack scenario, the attacker now forwards the delayed *Client request* intercepted from the valid client to the server over the TLS protected link that has been established by the attacker with the server in response to the TLS renegotiation request. The server accepts and interprets the forwarded client message as valid part of the renegotiation phase. Note that the response to a *Start Renegotiation* message is the same message as used for the initial connection attempt – the *Client* request. During the renegotiation phase the server will require client side authentication in a subsequent message. This enables an end-to-end key negotiation between the client and the server. So a valid client authentication is performed. All subsequent messages are now secured end-to-end and the attacker is not longer able to access them. But then, the stored *EVIL*-request that has been sent by the attacker is executed by the server with the permissions of the authenticated client.

This attack shows on the one hand a weakness of the TLS protocol due to the missing cryptographic session binding of the two TLS sessions, i.e. the one established before performing the session renegotiation, and the one established as result of the session renegotiation performing also client authentication. If there would be a session binding, the Web server would realize that the *Client requests* does not refer to a former session in which the original client was not involved. The consequence is that the Web server is in fact communicating with two different entities (the attacker and the client), while it assumes that it is communicating only with a single entity. On the other hand, the attack shows the insufficient integration of TLS into the application, as the web server in this example should have requested an affirmation of the *EVIL* request over the renegotiated TLS session before executing it. This weakness could be exploited for instance for stealing passwords or cookies from Web applications. The weakness has been addressed as part of an update of the TLS protocol using a binding of the initial session to the renegotiated session in the *ClientHello* and *ServerHello* messages [8]. To achieve this, client and server store the individually sent *verify\_data* from the *Finished* message of the previous handshake and reuse this

information in the *ClientHello* resp. *ServerHello* of the ongoing renegotiation handshake.

## V. CHALLENGE BINDING (PRE-CHALLENGE)

The problem of the man-in-the-middle susceptibility of simple challenge-response component authentication originates from the fact that the same component authentication mechanism or the same associated authentication key respectively is used in different usage contexts. Following common security design, different keys would be used for different purposes. Furthermore the cryptographic material should be bound to the intended usage context (i.e., to derive context-bound session keys from the response).

As in important commercially available implementations of component authentication, the verifier needs access to the unmodified response; the response value cannot be modified practically. Therefore, challenge binding is proposed as countermeasure that can easily be integrated with existing component authentication mechanisms: When a remote verifier cannot select an arbitrary challenge value, it cannot use the authenticating object as oracle to determine responses for an arbitrary received challenge value.

### A. Challenge Binding

The basic idea of challenge binding is to use a derived challenge value (bound challenge), which in turn is derived from a challenge value selected by a verifier (pre-challenge). For this bound challenge the response is calculated. The derived challenge is bound to a verifier by using an information of the verifier from which the (pre-)challenge has been received as derivation parameter. The (pre-)challenge selected by a verifier is thereby bound to the verifier context. This binding operation can be performed by the authenticated object itself or by a (trusted) intermediary node, see Figure 5.

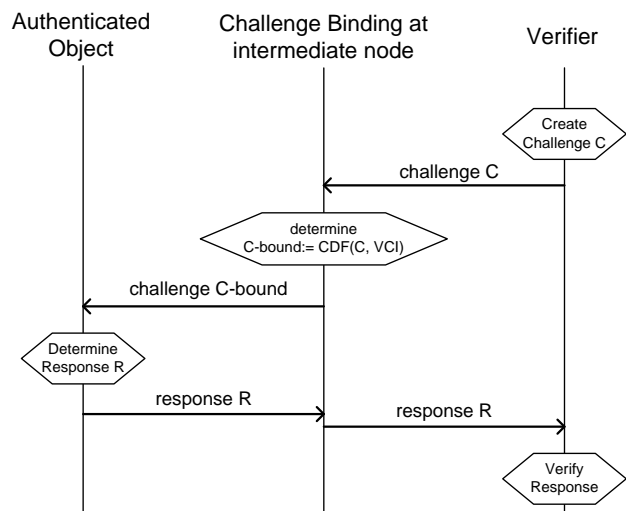


Figure 5. Challenge Binding

Note that the trusted intermediate node should be physically close to the authenticating object to avoid the

already described Man-in-the-Middle attack between the object and the intermediary node.

The challenge  $C$  (pre-challenge) selected by the verifier is sent to the object directly or to an intermediary node in close vicinity of the object to be authenticated (e.g., a control unit to which a sensor or actuator is directly connected), see Figure 5. The challenge derivation can be performed by both, the object to be authenticated itself, or by a trusted intermediary node. For the following description, the challenge binding is performed by an intermediate (trusted) component. This challenge  $C$  is now modified by deriving a bound challenge value  $C$ -bound using a non-invertible function (challenge derivation function, CDF). Verifier-dependent context information (VCI) is used as derivation parameter to bind the challenge to the respective verifier. In particular, the verifier's network address, node identifier, or a session key established between the verifier and the intermediary can be used.

This modified, verifier-context bound challenge  $C$ -bound is forwarded to the object to be authenticated. The object determines the corresponding response and sends it back to the intermediary that forwards the response to the (remote) verifier. The verifier determines the bound challenge  $C$ -bound as well, using the selected challenge  $C$  and the verifier context information VCI. Note that the VCI can be determined either by the verifier and the intermediary, if both are configured in a way to determine the VCI on available information (like certain address information of the verifier, see also section B below). Alternatively, the VCI may be sent as part of the communication from the intermediary or the verifier.

The remote verifying party can therefore not freely select the challenge for which a response is computed. Anyhow, it can be sure about the freshness of the challenge  $C$ -bound for which it received the response as it depends on the pre-challenge  $C$  selected by the verifier.

### B. Verifier Context Information

Verifier dependent context information is used as derivation parameter to bind the challenge to the respective verifier. There is a variety of parameters that can be used to specify a verifier context. In particular, the verifier identity, e.g., IP or MAC address, DNS name or URL, an (unpredictable) session ID, or a digital certificate or security assertion may be used. This information can be determined by the intermediary, without direct involvement of the verifier. This verifier context is used as parameter to separate two different verifiers. So in practice it must not be possible for a verifier to act successfully within a verification context belonging to a different verifier.

### C. Challenge Derivation Function

Requirements on a challenge derivation function are similar as for a key derivation function, i.e. being non-invertible and pre-image resistant (see [8] and [9] for more specific information on key derivation functions). Therefore, the functions that are typically used for key derivation can be used as challenge derivation function as well. For example, the bound challenge  $C$ -bound could be derived as HMAC-

SHA1( $C$ , VCI), using the challenge instead of a key, and using VCI as textual string determining the verifier context. Alternative key derivation functions may be the higher SHA methods like SHA256 or SHA512 in combination with the HMAC or symmetric algorithms like the AES in CBC-MAC mode [10] or in GMAC mode [19] as already noted in the overview of section II.

A further approach to be named here is the application of key wrapping as described for instance in [20] using AES. Here, the challenge could be used to derive an encryption key for the key wrapping algorithm. The information encrypted includes additional information provided by the verifier and generated by the authenticating component. Both pieces of information (encrypted part and generated part) need to be sent to the verifier. By decrypting this information with the key derived from the challenge, the verifier can proof if the decrypted content equals with the generated content. In the positive case, the component was successfully authenticated. This is depicted in the following figure.

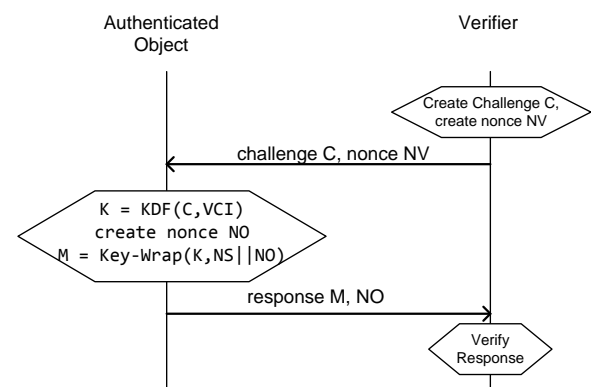


Figure 6. Application of Key Wrapping for Challenge Binding

General recommendations for key derivation functions using pseudorandom numbers are also provided in [9].

## VI. APPLICATION TO IP-BASED SMART OBJECTS

One possible application of protected remote component identification is IP-based communication within the Internet of Things. A node communicating with a smart object ("thing") over IP-based networks wants to verify the identity of the smart object or of a component being part of or being integrated into the smart object. Communication can be realized e.g., using HTTP-based Web Service protected by TLS or by IP-based communication protected by IPsec. A challenge-response based smart object authentication can be integrated in well-known protected communication protocols, as HTTP Digest over unilaterally authenticated SSL/TLS, or EAP (Enhanced Authentication Protocol), or within IKEv2 for IPsec.

However, the challenge is modified using verifier context information as derivation parameter. Here, besides the nodes identifier (server name resp. IP address) also the purpose and the used communication protocol can be used as challenge derivation parameter (e.g., "*DeviceComponent-Authentication/HTTP-DIGEST/TLS*" || *Server-IP*).



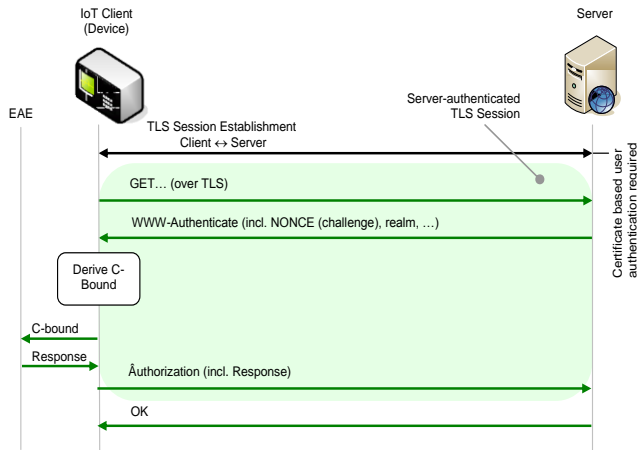


Figure 7. IoT Device Authentication Using HTTP DIGEST

Figure 6 shows an exemplary use for Internet of Things (IoT) device authentication. An IoT device includes a low-cost object authentication IC as embedded authentication element (EAE). The EAE realizes only a simple challenge response authentication mechanism. When the IoT device communicates with another device, e.g., a server, a server-authenticated TLS session is established. The device authenticates itself over HTTP using a modified HTTP digest. From the received NONCE value, i.e. the challenge value, a bound challenge value C-Bound is derived. The derivation parameters can include, besides the received realm, also further context information of the session, e.g. server IP address, server DNS name, server public key hash or server digital certificate hash, or the common name included in the server certificate. Also, the approach described at the end of section IV for TLS can be leveraged directly. This would result in the binding to the actual TLS session context, over which the authentication is being performed. The bound challenge is sent internally of the IoT device to the embedded authentication element EAE included in the IoT device. The received response is provided to the server for verification. While TLS has been used as example, also different protocols as datagram TLS (DTLS) or network / link layer security could be used as well.

The described IoT device authentication using the embedded authentication element may be used for device authentication during regular operation. Advantageous is, however, its use for automatically setting up the initial device configuration after the IoT device has been installed. This automatic security bootstrapping allows the IoT device to authenticate towards a bootstrapping server that provides security configuration data that is then used during the operational phase. Moreover, this bootstrapping server may additionally provide the functionality of a secure inventory management, which may need to be contacted by the IoT device in regular intervals. Thus, the server can check the component authenticity also regularly.

## VII. RELATED WORK

Most similar to our proposal is the binding of an authentication challenge for a PUF authentication to the hash of the requesting program, see [11] the verifier selects a pre-challenge, from which a bound challenge is derived using the hash of the verifier program as input to the challenge derivation. Note that the binding to the hash of the verifier program alone, without address information is weaker, as the hash is supposed to be the same on different hosts. Thus, an attacker possessing the verifier program may still perform the attacks described in section II.

The insecurity of tunneled authentication protocols has been analyzed [12]. In real-world environments, often an existing security deployment and authentication shall be re-used for a different purpose. In particular tunneled EAP authentication was considered, e.g., based on PEAP (Protected EAP). The described countermeasure was binding cryptographically the results (session keys) of the two authentication runs, i.e., the inner and the outer authentication, or by binding the session key to an endpoint identifier.

Performing a key derivation is a basic building block for designing secure communication. Various required session keys can be derived from a common master session key. NIST recommended a key derivation function, using a usage-describing textual string as derivation parameter [8]. Another example is the pseudorandom function used within TLS [7], which uses secret keys, seeds and textual strings (identifying label) as input and produces an output of arbitrary length. The same approach is taken in the Multimedia Internet Keying MIKEY [13].

It is also known to bind an authentication to properties of the used communication channel [14]. Two end-points authenticate at one network layer and bind the result to channel properties to prevent against man-in-the-middle attacks where the attack would result in different channel binding properties from the viewpoint of the authenticating nodes.

Furthermore, non-interactive key agreement schemes allow to derive a common, shared key material between nodes that have received a key bound to the own identity [15]. No protocol exchange is required to derive this shared key, but the key is derived similar as with a key derivation function. However, the two derivation steps for binding a root key to two node identifiers can be performed commutatively.

In the “cuckoo attack”, an attacker causes a user of a computer including a trusted platform module (TPM) that a different TPM that the adversary controls resides in the user’s computer [16]. The adversary’s TPM can therefore make false assertions about the software running on the user’s computer. This attack is an example of a man-in-the-middle attack where the adversary sits between the verifier and the TPM.

Another option to bind an initial authentication to further challenges and authorizations tokens is the Kerberos, a well established security protocol, which has native support in several operating systems. The general proceeding of

Kerberos is described in [18] and depicted in Figure 8. In the target use case, the Kerberos server resembles the binding of an authorization request to a dedicated environment. It performs as a challenge response three party protocol, while the solution described in section V binds the challenge “on the way” to the object without having a three party (challenge-response) protocol. Furthermore, the solution proposed in this paper assumes physical closeness of the binding node to the actual object to be verified.

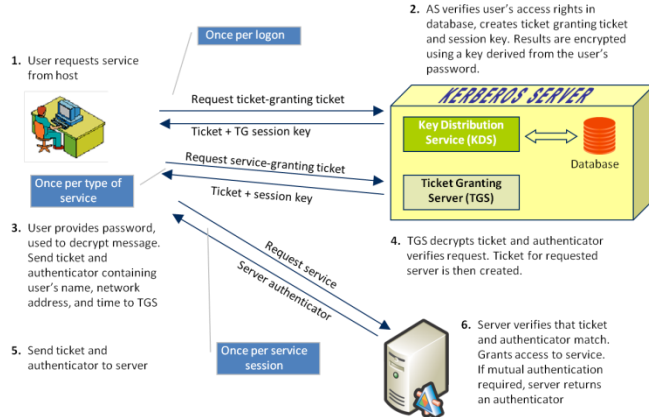


Figure 8. Kerberos Protocol Overview

An advantage of the proposed solution is that it uses less message exchanges between the participating nodes to bind the authentication to a dedicated environment compared to Kerberos stated above.

## VIII. SUMMARY AND OUTLOOK

An attack on component authentication has been described where a single genuine component is used as oracle to compute valid authentication responses. A single malicious verifier may use an obtained valid response value to authenticate as the genuine component towards other verifiers. The described attack is made possible by the fact that the cryptographic solution for component authentication is used within a different usage environment than it has been designed for: The attack is relevant when the component authentication for verifying the genuineness of a component is performed not only locally, but also remotely. This attack is also an example that a small functional enhancement – here making an existing functionality accessible remotely – can have severe implications on security.

This paper proposed a challenge binding mechanism as countermeasure for the described attack. The available, extremely cost-efficient object authentication technology can thereby be used securely also for a different purpose than the one it has been designed for originally. An authentication challenge is bound to the verifier so that a remote verifier can neither simulate a local, unbound authentication nor can it simulate an authentication towards a different remote verifier having a different associated verification context. A possible application of this general challenge-binding mechanism is the cost-efficient authentication of devices within the Internet of Things.

## REFERENCES

- [1] R. Falk, S. Fries: Protecting Remote Component Authentication, The Fifth International Conference on Emerging Security Information, Systems and Technologies SECURWARE2011, 21-27 Aug. 2011, Nice / Saint Laurent du Var
- [2] Atmel CryptoAuthentication, <http://www.atmel.com/products/cryptoauthentication/>, last access Jan. 2012
- [3] Atmel CryptoAuthentication Product Uses, Application Note, 2009. [http://www.atmel.com/dyn/resources/prod\\_documents/doc8663.pdf](http://www.atmel.com/dyn/resources/prod_documents/doc8663.pdf), last access Jan. 2012
- [4] Infineon ORIGA, <http://www.infineon.com/ORIGA>, last access Jan. 2012
- [5] Verayo <http://www.verayo.com/products/unclonable-rfids>, last access Jan. 2012
- [6] Atmel CryptoAuthentication High level Security Models, Application note, 2009. [http://www.atmel.com/dyn/resources/prod\\_documents/doc8666.pdf](http://www.atmel.com/dyn/resources/prod_documents/doc8666.pdf), last access Jan. 2012
- [7] T. Dierks and E. Rescorla: The Transport Layer Security (TLS) Protocol Version 1.2, RFC 5246, August 2008
- [8] E. Rescorla, M. Ray, S. Dispensa, and N. Oskov: Transport Layer Security (TLS) Renegotiation Indication Extension, RFC 5746, February 2010
- [9] Lily Chen: Recommendation for Key Derivation Using Pseudorandom Functions, NIST Special Publication 800-108, 2009, <http://csrc.nist.gov/publications/nistpubs/800-108/sp800-108.pdf>, last access Jan. 2012
- [10] John Black and Philipp Rogaway: A Suggestion for Handling Arbitrary-Length Messages with the CBC MAC, <http://www.cs.ucdavis.edu/~rogaway/papers/xcbc.pdf>, last access Jan. 2012
- [11] Srinivas Devadas: Physical Unclonable Functions and Applications, Presentation Slides (online), <http://people.csail.mit.edu/rudolph/Teaching/Lectures/Security/Lecture-Security-PUFs-2.pdf>, last access Jan. 2012
- [12] N. Asokan, Valtteri Niemi, and Kaisa Nyberg: Man-in-the-Middle in Tunnelled Authentication Protocols, LNCS3364, Springer, 2005.
- [13] J. Arkkom, E. Carrara, F. Lindholm, M. Naslund, and K. Norrman: MIKEY: Multimedia Internet KEYing, RFC3830, August 2004
- [14] N. Williams: On the Use of Channel Bindings to Secure Channels, Internet RFC5056, 2007
- [15] Rosario Gennaro, Shai Halevi, Hugo Krawczyk, Tal Rabin, Steffen Reidt, and Stephen D. Wolthusen: Strongly-Resilient and Non-Interactive Hierarchical Key-Agreement in MANETs, Cryptology ePrint Archive, 2008. <http://eprint.iacr.org/2008/308.pdf>, last access Jan. 2012
- [16] Bryan Parno: Bootstrapping Trust in a Trusted Platform, 3<sup>rd</sup> USENIX Workshop on Hot Topics in Security, July 2008, [http://www.usenix.org/event/hotsec08/tech/full\\_papers/parno/parno\\_h.html](http://www.usenix.org/event/hotsec08/tech/full_papers/parno/parno_h.html), last access Jan. 2012
- [17] M. Braun, E. Hess, and B. Meyer, “Using Elliptic Curves on RFID Tags,” International Journal of Computer Science and Network Security, Volume 2, pages 1-9, February 2008
- [18] C. Neuman, S. Hartman, K. Raeburn: RFC4120: The Kerberos Network Authentication Service, July 2005
- [19] Morris Dworkin: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, NIST SP 800-38D, November 2007, <http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf>, last access June 2012
- [20] NIST, AES Key Wrap Specification, November 2001, [http://csrc.nist.gov/groups/ST/toolkit/documents/kms/AES\\_key\\_wrap.pdf](http://csrc.nist.gov/groups/ST/toolkit/documents/kms/AES_key_wrap.pdf), last access June 2012

# A Privacy Preserving Solution for Webmail Systems with Searchable Encryption

Karthick Ramachandran, Hanan Lutfiyya and Mark Perry

Department of Computer Science

University of Western Ontario

London, Ontario, Canada

Email: {kramach, hanan, markp}@csd.uwo.ca

**Abstract**—In this work, we give an introduction to privacy issues in Cloud Computing and discuss the state of art in the privacy enhancing technologies that can be used for Cloud Computing. We focus on a Software as a Cloud scenario (webmail services) and propose a privacy preserving architecture in which users can retain their mail in the servers of their service providers in a cloud without compromising functionality (searchability of mails) or privacy. We benchmark our system and detail the results showing that it is feasible to architect a privacy preserving solution for webmail systems.

**Keywords**- *privacy-preserving, webmail, encrypted search.*

## I. INTRODUCTION

Cloud Computing is a model of computing in which the users can rent infrastructure, platform or software services from other vendors without requiring the physical access to the rented service [1]. There are three main types of cloud offerings (Figure 1): Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). IaaS offers virtualized instances of bare machines leaving the installation and customization of softwares including the Operating System to cloud computing customers (eg.: Amazon, Rackspace, Slicehost). In PaaS, an application framework is provided to the customers for developers to develop their software with (eg.: Google App Engine, Microsoft Azure). A SaaS provider offers a particular application as a web service, which customers can customize to their needs (eg.: Google Docs, Salesforce etc). The Cloud Service Provider (CSP) focuses on infrastructure and software expertise and aims to optimize their utility by providing centralized services for one or many clients. The benefit to the cloud service client (CSC) is that the cost associated with the underlying infrastructure and software services needed to support the CSC's application is reduced. There are two reasons for the cost reduction. One reason is that the underlying infrastructure and software services are shared among CSCs. The second reason is that since a CSP manages data, it can use creative business models like Contextual Advertising Model [2] for generating revenue by delivering advertisements to users based on the data. For example, webmail services such as Google can provide Gmail for free. As a result, Cloud Computing has been widely adopted. MarketsandMarkets [3] estimates that the

cloud computing global market will increase from \$12.1 billion (US) to \$37.8 billion (US) in 2015 at a compound annual growth rate of 26.2 percent.

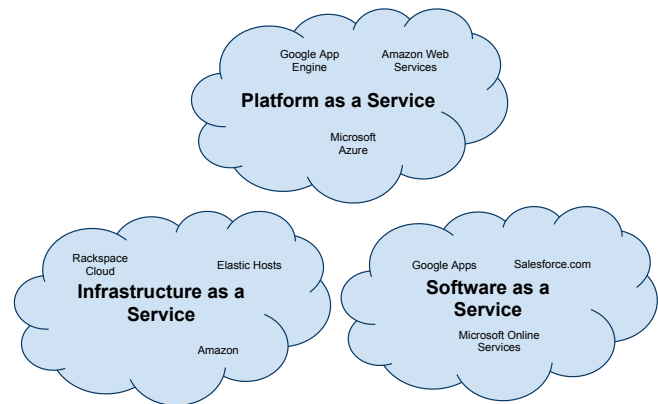


Figure 1. Cloud Architectures

In spite of this widespread adoption, organizations are still wary of storing their sensitive data with a CSP. Privacy risk remains a major concern in the cloud computing environment [4].

The definition of privacy that we use was defined by Warren et al. [5] in 1890. Warren et al. described privacy as the “right to be let alone” with the focus on protecting individuals. The Universal Declaration of Human Rights states that “No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honor and reputation. Everyone has the right to the protection of the law against such interference or attacks.” [6]. Modern legislation encompasses these ideas – privacy is the need to protect and control information about the individual by that individual.

There are a variety of ways that the privacy of data can be compromised in a cloud service environment [7]. This includes the following:

1) *Sharing of data with an unauthorized party*: The Cloud provider could compromise the confidentiality of the data by sharing the data that it stores with unauthorized parties. This can go against the terms and conditions of the

service and will qualify as a breach of security and contract. The end user may never be aware of such a breach.

2) *Corruption of data stored:* The Cloud Computing provider's root access to physical machines allows the Cloud Provider to have access that allows the Cloud Provider to modify/delete data. The Cloud Provider could tamper with the data making the data non-usable or modify the data in a way that system cannot detect the modification. This poses a serious threat to the integrity of the application.

3) *Malicious Internal Users:* The employee of a Cloud Computing Provider who has root access to these physical machines, could access the data and use it for their own advantage.

4) *Data Loss or Leakage:* When a virtual machine is used in an infrastructure, it poses a variety of security issues [8], which could lead to a compromise of the data. Moreover, when the facility that hosts the user's data is subjected to a natural calamity, it could risk the loss of the user's data.

5) *Account or Service Hijacking:* Another risk for the Cloud Computing provider is, if the service is hijacked, or the computer is hacked into by an intruder, the hacker will have access to data.

Storing the data in the cloud, can increase the privacy risks for the following stake holders:

- 1) Cloud Computing User
- 2) Organization using the Cloud Service
- 3) Implementors of Cloud Platforms
- 4) Providers of application on top of cloud platforms
- 5) For the data subject

This work focuses on the following threats: (a) Sharing with an unauthorized party, (b) Malicious internal users, and (c) Account or service hijacking. Our work applies to the class of cloud services that stores data and provide searching as its primary functionality. This includes services such as webmail, collaborative document authoring (Google documents) and private blogs. The example used throughout this paper is webmail.

We proposed Chaavi [9], a webmail infrastructure that builds on the public/private key model to encrypt email with a custom implementation of encrypted indices for keyword searches using the server's infrastructure. Chaavi is the first system that addresses the above threats in a real working environment.

The rest of paper is organized as following. A motivating example of webmail services is described in Section II. Section III presents some of state of the art in preserving privacy for cloud computing services. Section IV reviews background and related work for searching on encrypted data. Section V presents the architecture of Chaavi system. The implementation details are discussed in Section VI. Section VII presents the experiments conducted to study the system and we conclude by stating our contribution and future work in Section VIII.

## II. MOTIVATING EXAMPLE: WEBMAIL SERVICES

Webmail services offer user convenience. A username, password, and Internet access users, are not tied to any particular equipment or location. Webmail services primarily offer the following functionality:

- 1) Mail Storage
- 2) Organization of mail
- 3) Keyword Searching

For (1) and (2), the service provider need not know the exact content of the mail. However, for performing a plain-text keyword search on email the user needs the service provider to know the content of the mail, so that the cloud provider's infrastructure can be used to index the mail content, which can in turn be used for the search process.

The usage of webmail services, has the following shortcomings:

- 1) The need to trust the service provider (e.g., Google, Yahoo, or Microsoft) as the mail is stored as plain-text in the service providers' servers (or using single key encryption). The mail is then prone to insider attacks (anyone with the access control will be able to read the mails).
- 2) There is an assumption that the provider is honest, and the security level is sufficient.
- 3) When the mail is transferred from one domain to another, it is transmitted through SMTP [10]. SMTP as a protocol does not support encryption. Technologies like Transport Layer Security [11] are used to transfer mail to other domains. However, the data is still protected only up to the layer at which it reaches the target mail server. Once it reaches the target mail server, the mail is again prone to insider attacks in the new domain.

To address such problems, various client encryption systems, such as Pretty Good Privacy (PGP) [12], have been developed. However, encryption using PGP make the mail non-searchable in the web server.

## III. RELATED WORK

Privacy Enhancing Technologies (PET) can be used by the developers of the application to enhance the individuals privacy in an application development environment. In this section, we survey state of the art in PET.

PET technologies include:

- 1) Privacy management tools that enable inspection of server-side policies that specify the permissible accesses to data
- 2) Secure online access mechanisms to enable individuals to check and update the accuracy of their personal data
- 3) Anonymizer tools, which will help users from revealing their true identity by not revealing the PII (Privately Identifiable Information) to the cloud service provider.



**Homomorphic Functions:** Homomorphic encryption schemes refer to asymmetric encryption techniques, where algebraic operations on plain text can be performed directly on a respective cipher text. This was first introduced by Goldwasser et al. [13], where the authors performed modular addition of two bits using multiplication of ciphertexts. The two kinds of homomorphic functions are the following:

- 1) Partially homomorphic functions and
- 2) Fully homomorphic functions

Partial homomorphic functions enable either addition or multiplication on plaintexts. However, in a fully homomorphic scheme, both operations are supported. Fully homomorphic functions, allow executions of programs in untrusted party without revealing the input to the party. The untrusted party can be seen as a cloud provider.

Craig Gentry [14] described the first fully homomorphic encryption scheme based on lattice-based cryptography. However performing google search on encrypted keywords using homomorphic encryption based on Gentry's scheme will increase the computing time by trillion.

Homomorphic encryption remains in the theoretical realm as more advanced abstractions need to be created for using homomorphic functions in practical applications.

**Privacy By Secure Computation:** The objective of secure computation is to evaluate a function  $f$  that takes inputs from two parties A and B without revealing the exact inputs to each other. The Yao's protocol [15] provides some of the basic techniques to perform a computation in a secure way without revealing the inputs. The Yao's protocol forces the expression of a computation problem in terms of logical circuit using gates. The input of each gate is randomly encrypted and then the final resulting output is decrypted to get the exact answer of the computation. The encryption and the decryption is done at the client's end. The expression of a simple problem using the Yao's protocol is found to be complex. Applications that typically reside in the cloud (e.g., mail) are too complex for this.

**Privacy By Using Secure CoProcessors:** Secure co-processors are currently the only realistic way to perform general-computing even when an adversary has direct physical access to the server. In our case the adversary could be the cloud service provider itself. It is a very limited computer with ROM, RAM and battery backup for persistent storage and an ethernet card. When installed in a computer, co-processors can be seen as a secure area inside a computer, which even the main processor cannot access. Privacy as a Service [16] recognizes these factors and proposes a system architecture in which a coprocessor is installed in every Cloud Computing system. The data loaded into the cloud is classified based on its significance and security by the cloud user (No Privacy, Privacy with Trusted Provider, Privacy with Non-Trusted Provider). The data tagged with Privacy with Non-Trusted Provider level is processed by the secure co processor.

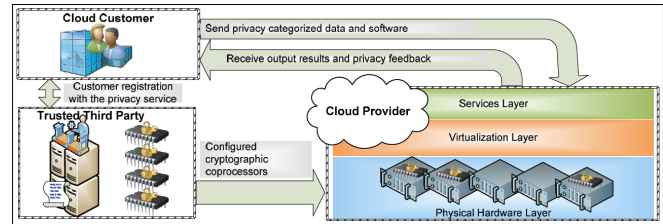


Figure 2. System Model for Privacy by Secure CoProcessors [16]

Figure 2 [16] is an example of a system built using secure coprocessors. Cloud customers, Trusted Third Party and the Cloud Provider are the three main stakeholders of this system. The coprocessor is signed by secret keys by the trusted third party and then is supplied to cloud provider. When a new customer registers with the cloud provider, they share the secret keys with the trusted third party. The co-processors can directly contact the trusted third party for the keys to encrypt the secret data within the coprocessor. The data channel between the co-processor and the trusted third party is secured using a mutually agreed upon public/private key pair during the initial time of supply of co-processors to CSP by trusted third party. Secure co-processors need a separate hardware installation in server. Also co-processors are expensive and are not yet economical to be used in a cloud computing environment.

Trusted Program Module (TPM) is a secure cryptoprocessor specification introduced by Trusted Computing Group to standardize the usage of crypto coprocessors [17]. TPM chips can be used to attest platform integrity, to enable disk encryption and for secure storage of selective sensitive entities such as the username and password. It provides basic function of RSA 2048 bit public key cryptography protected by hardware. According to Wave Systems Corp. [18], more than 350 million PC's are shipped with TPM as of 2010. TPMs introduce high overhead in the execution of an application [19]. However Jonathan et.al [19] argue that, as the usage of TPMs get popular, the future hardware performance will improve.

**Privacy By Encryption:** Privacy can be enforced by encrypting all the data that is stored in the cloud. The main issue is that the cloud can be only used for storage of the data. As the data will be unrecognizable to the cloud service provider, it will not be possible for the cloud service provider to process the data nor to perform some number crunching tasks. Searchable encryption uses an algorithm, which allows users to encrypt the data and then provides the server with trapdoor information [20], so that the server can search for a given string through the searchable encryption algorithm. This part is discussed in detail in Section IV-C.

Privacy-Preserving Multi-keyword Ranked Search over Encrypted Cloud Data [21] proposes a new encryption scheme for keyword search over encrypted data in cloud

computing environment with privacy and performance requirements.

In our work we achieve privacy by encryption by using searchable encryption scheme for a webmail software. Our focus is to study how the encryption schemes can be engineered in a real working environment. This is an extension of our previous work [9] with more details on related work, implementation and conclusion.

#### IV. BACKGROUND

In this section, we review the basic elements common to webmail infrastructures. We also present an introduction to PGP and searchable encryption.

##### A. Mail Architecture

The webmail infrastructure is responsible for end to end delivery of email. Figure 3 presents architectural components and protocols typically used to support webmail applications.

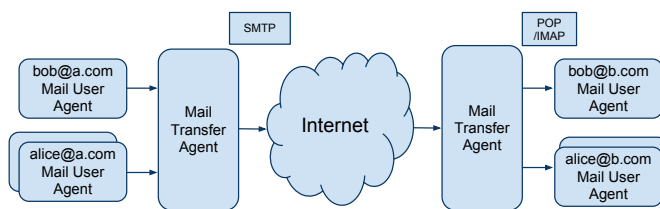


Figure 3. Email Architecture

1) *Components*: This subsection describes the architectural components.

*Mail User Agent*: The Mail User Agent (MUA) is used to manage a user's email. It acts on behalf of the user to send and receive mail from the Mail Transfer Agent (MTA). Popular MUAs include Microsoft Outlook, Mozilla Thunderbird, Apple Mail. In a webmail system, the MUA runs in the server and the pages are rendered as HTML pages for the browser.

*Mail Transfer Agent*: The Mail Transfer Agent (MTA) transfers messages from one server to another. It receives email either from another MTA or MUA. The transmission of email follows standardized protocols for message transfers.

2) *Protocols*: This subsection describes commonly used protocols.

*Simple Mail Transfer Protocol (SMTP)*: SMTP refers to the standard for the transfer of messages from one server to another. It is used by MUA to relay mail through MTA and it is also used by MTA to send and receive mail between other MTAs. SMTP as a standard does not encrypt messages (unless Transport Layer Security encryption is used).

*Post Office Protocol (POP) / Internet Mail Access Protocol (IMAP)*: POP/IMAP are email retrieval protocols that specify standards for downloading messages from the MTA for MUA. Examples of use is found with support for POP version 3 and IMAP as provided by Gmail.

3) *Privacy Threats*: In webmail systems, there is a server for webmail introduced into the standard mail system (Figure 3). It acts as the Mail User Agent for a number of users and manages email for all the users. The MUA, unlike the standard model (Figure 3), is centralized at the server. The webmail server uses POP/IMAP to download messages from MTA.

There are several privacy concerns with respect to email systems. If the connection to the webmail server is not secured using Hypertext Transfer Protocol Secure (HTTPS) all the data between a user's browser and the server will be in plain text. SMTP, unless used with Transport Layer Security (TLS) layer, is insecure. Even if the TLS layer is used, the mail will still be accessible by the owner of the MTA, through which the mail is routed. This is because TLS is designed to protect data in an insecure network (like Internet) and not from the communicating parties. Some of the security threats involved in email systems are identified by Kangas et al. [22], and Kaufman et al. [23]. These are detailed below.

*Eavesdropping*: When email is unencrypted, potential hackers who have access to network packets flowing through the network will be able to read the email sent. This can be achieved by enabling the promiscuous mode on ethernet cards.

*Identity Theft*: If the user's username and password is obtained, then hackers have full access to all the email content. Such password information can be obtained by eavesdropping on the network.

*Invasion of Privacy*: The recipient of the mail is able to get more information from the email header information than what the sender intends to reveal. For example, the header will reveal the sender's SMTP IP address and subject of the email sent.

*Message Modification*: Anyone who has administrator access to the webmail server can modify the messages stored in the server. It is not always possible for a recipient to determine that email has been tampered with.

*False Messages*: It is relatively easy to create false messages and send it as if it is from any person (as evidenced by spam).

*Message Replay*: Akin to message modification, the message created by user can be saved and sent again and again.

*Unprotected Backups*: Messages are stored in plain-text on SMTP servers, and backups will also contain complete copies of the messages. Even when the user deletes a message from the server, the backup will still hold the content.

**Repudiation:** As email messages can be forged (for example see your spam box), there is no way of validating that the email has been in-fact sent by a particular person. This has serious implications in business communications, electronic commerce.

### B. Pretty Good Privacy

PGP was created by Zimmermann et al. [12], in 1991 to address the security issues with email. PGP encryption uses a serial combination of hashing, data compression, symmetric-key cryptography, and public-key cryptography. Each public-key is bound to an email address. It serves as the verification mechanism for the origin of the email. As the email is encrypted using the private key of the user and the encrypted version is sent into the network, it addresses many security issues of the email infrastructure. For webmail systems, software such as FireGPG [24] provide a browser extension that implements PGP. As PGP support enhances the security of the email system by encrypting the mails, the mail becomes unreadable by server. Hence the server cannot perform keyword searches on the mail.

### C. Searchable Encrypted Data

Public Key Encryption with Keyword Search (PEKS) [20] is one of the seminal works in the area of making encrypted data searchable. The authors of PEKS propose to encrypt the message using the Public-Private key infrastructure. Along with this cipher text a Public-Key Encryption with Keyword Search (PEKS) of each keyword (the words that make up the message) is appended to the final message. To send a message  $M$  with keywords  $W_1, W_2, \dots, W_m$  the following information is transmitted to the server:

$$E_{A_{pub}}(M) \parallel PEKS(A_{pub}, W_1) \parallel \dots \parallel PEKS(A_{pub}, W_m)$$

where  $A_{pub}$  is the public key of the user,  $E_{A_{pub}}(M)$  is the encrypted message,  $PEKS$  is the function that encrypts the keywords using  $A_{pub}$ . To test whether a word  $W$  is a part of the message, a user supplies  $PEKS(A_{pub}, W)$  along with a trapdoor function  $T_w$  to the server, that can test whether  $W = W'$  ( $W'$  being the keywords that are stored in the encrypted form in the server). If  $W \neq W'$  the server learns nothing more about  $W'$ .

Public Key Encryption with Keyword Search Revisited [25] identifies some of the issues with the original PEKS and proposed a provably secure algorithm. The authors argue that if in PEKS the server starts learning the trapdoor then there can be a categorization of mail formed just based on the learned trapdoor information. The trapdoor information is the extra information sent to the server along with the encrypted keyword for the server to test for the existence of a keyword.

The authors also identify that in PEKS there is an assumption that the communication channel between the sender and the server is secure. To enable secure communication

through insecure channels the authors propose a Secure Channel Free Public Key Encryption with Keyword Search (SCF-PEKS), that uses a server's public-private key pair for communication.

## V. ARCHITECTURE

This section describes the various components of Chaavi. Figure 4 gives the overall architecture of the system.

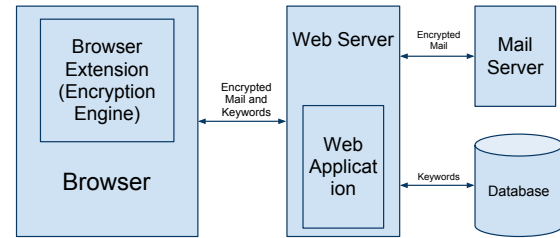


Figure 4. Chaavi - Architecture

### A. Browser

The browser is responsible for rendering the pages created by the web application. Its default behavior can be modified or enhanced by using extensions or plugins in the browsers. Modern browsers such as Mozilla Firefox, Google Chrome provide functionality to write extensions/plugins and install the extensions locally.

### B. Browser Extension

A browser extension is used in Chaavi to encrypt the secure message sent to the server. It is also used to decrypt the messages that are sent from the server. Additionally it has key generation and key management functionality. The extension is composed of the following modules.

**Public-Private Key Generation:** As stated earlier, Chaavi uses a public/private key model for securely communicating messages. In a public/private key model, a public-private key pair is generated when the system is initiated for the first time, for a particular user. The messages encrypted by the public key can be decrypted only by use of the private key. The public key as the name implies is shared in a public forum.

**Keyword Encryption Key Generation:** Public-Private key pair is used for secure message communication. A symmetric key is also generated to encrypt the individual keywords present in the mail. A symmetric algorithm (unlike the Public-Private key) is used here as the keywords need not be decrypted by anyone else other than the sender of the message.

**Key Management:** Key management is performed using a graphical user interface (GUI). The GUI enables the user to add or delete the public keys of the recipients with whom the user wants to communicate through mails.

**Encryption:** The functionality of the encryption module is to encrypt the messages that are sent to the server from the browser. It also extracts and encrypts the individual keywords in the message. The encryption module is triggered from the web application when the user submits a mail to send it to the web server. This module encrypts the message using the recipients's public key and the keywords with the keyword encryption key.

**Decryption:** When an encrypted message is sent from the server to the browser, the decryption module decrypts the messages using the private key of the user that is generated during system initialization.

### C. Web Application

The webmail application provides graphical user interfaces for the users to read, send and search messages. It comprises of both server-side and client-side (browser) functionality.

When a user sends a message from the web application (Figure 5), the Encryption module encrypts the message and extracts and encrypts the keywords. The web application sends the encrypted message and keywords to the web server. On receiving the encrypted message and the keywords, at the server-side the application saves the encrypted message alongside the encrypted keywords in a database for future retrieval. The application then transfers the mail to the Mail Server (SMTP server) for the mail to be delivered to recipient.

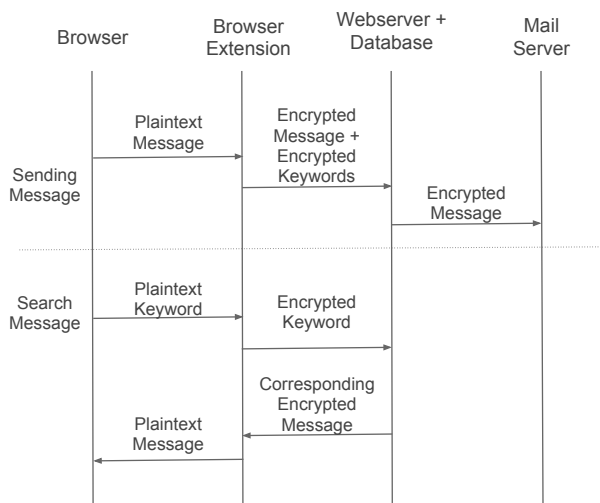


Figure 5. Sending and Searching for a Message

When the user wants to search for a particular keyword in their inbox, the encrypted keyword is sent to the server-side. The web application then searches for the mails corresponding to that particular encrypted word and then sends the encrypted mails back to the user.

### D. Database

The mail storage and organizational functionality is already handled by the web application provided by Squirrelmail. One custom table, *search* is added to the database, which stores the  $\langle message\_id, encrypted\_keyword \rangle$  pair. This database is looked up when the user performs a keyword search.

### E. Mail Server

The mail server sends and receives email communicated to it through the Internet. The mail server functionality is not modified by our system. The web application communicates with the mail server to send and receive messages.

## VI. IMPLEMENTATION

The following software is used to implement the different components in the system:

- Browser - Google Chrome
- Browser Extension - Google Chrome using Javascript
- RSA encryption/decryption library from hanewin.net [26]
- AES encryption library [27]
- Web Application - Squirrelmail over PHP and MySQL
- Mail Server - Using the POP3 interface of the *csd.uwo.ca* mail server

The implementation details of individual modules of the system are detailed below.

#### A. Browser Extension

**Public-Private Key Generation:** The RSA algorithm [28] is used for the creation of keys. The key requires two large prime numbers as the input along with a random seed. All of these inputs are created by the extension randomly and provided as input for key generation. The keys are then stored locally along with the user name, for future retrieval in the local browser database. The key generation is implemented using the RSA libraries available from hanewin.net [26].

**AES Key Generation:** The symmetric AES key algorithm is used to encrypt the individual keywords present in the mail. The AES key generation algorithm takes as input a random seed, which is provided by requesting the user to move the mouse over the browser window. That generates some random co-ordinates, which is then used to generate the key.

AES is a natural choice for the symmetric key algorithm as it has been analyzed extensively and used worldwide [29]. However, unlike PEKS [25], AES algorithm does not support trapdoor and hence it is susceptible to chosen plaintext attacks (The attacker has the capability to choose arbitrary plaintext and the corresponding cipher texts). Moreover the encryption of the keywords under AES negates the possibility of performing range searches (e.g.,  $10 < b < 20$ ) or similarity searches (name starting with 'ka').



**Key Management:** The GUI for key management (Figure 6) is developed using the options functionality provided by the Chrome extension framework. It is used to insert the public keys of the recipients with whom the user wants to communicate. The private key of the user cannot be managed using this interface (the system automatically generates it when the user logs in for the first time). The keys are stored in the local storage database provided by HTML5. The local storage enables key-value storage locally managed by browser.

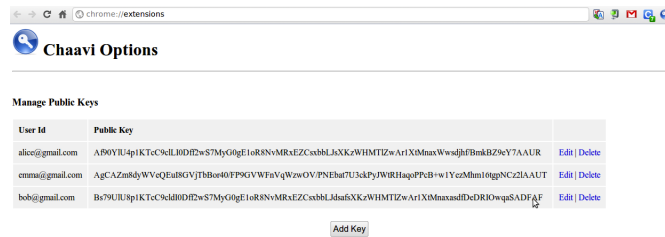


Figure 6. Key Management

**Encryption:** The user is provided with a HTML form from the web application, which contains input fields to enter the recipient email address, subject and the contents of the mail. The form submission event (*onsubmit* event) is associated with a custom submit event handler, which is hooked to the encryption module. The encryption module encrypts the contents of the mail using the user's public key and replaces the value in the field (contents of the mail) with the encrypted message. Along with this, the keywords in the message are extracted by the keyword extraction function and each keyword is encrypted using the AES key and stored in an object. The keyword extraction This object is serialized in JSON (Javascript Object Notation) and sent to the server along with the encrypted message.

**Decryption:** When an encrypted message is sent from the server to the browser the server adds the attribute value *post-deencrypt* to attribute *class*. The extension identifies these messages and decrypts the messages using the private key of the user. This decrypted message replaces the original encrypted message in the html page so that the user can see the message in the encrypted mail.

### B. Web Application

An open source web application (Squirrelmail) is identified and it is modified for our application. Squirrelmail is responsible for storage and organization of the mails. Our custom module is developed in PHP and added to Squirrelmail to save the encrypted messages alongside the encrypted keywords and for the retrieval of the messages based on the given encrypted keyword.

## VII. EXPERIMENTS

The performance of algorithms used in Chaavi (Privacy Preserving Web Mail with Keyword Searches) is studied in terms of space and time consumed by the algorithm in the local client system. Even though the performance of the encryption algorithms has been studied before, we focus on the performance of our system. The results presented in this section are intended to provide some insight on the overhead provided by the algorithms in a browser based extension environment. Since encryption and decryption is performed in the client browser system, the encryption and decryption is independent of the number of users currently using the system. Hence, we focus on the performance of the encryption algorithms for a browser-based extension environment.

All the experiments are executed in a Pentium IV Core 2 Duo processor using Google Chrome 5.0.375.99 beta.

### A. Time Complexity

The following algorithms are studied with respect to the execution time.

- Key Generation
- Encryption and Decryption (RSA Algorithm)
- Keyword Encryption (AES Algorithm)

**1) Key Generation:** Key generation is expensive since it involves finding two large random prime numbers and finding a product of the prime numbers based on the given random seed. The length of keys (as measured by bits) can be of sizes: 128, 256, 512, 1024. The higher the number of bits used, the more difficult it is to break the key (According to Schneier et al. [30], for breaking AES with key size greater than or equal to 256-bit through brute force will require fundamental breakthroughs in physics and understanding of universe). However, generating larger keys is time consuming. We present the average time taken for key generation for different bit sizes in Figure 7.

As can be seen the keyword bit size increases the creation time exponentially. The 1024 bit key generation takes around 41 seconds. However, as this is a one time activity (when the user sets up the system) the usability and inconvenience is minimal.

**2) Encryption and Decryption:** When the user wants to send an email the encryption module is executed each time, and the decryption module is activated when the user wants to read an email. This is a frequent activity and therefore more computation time spent on these modules will impact usability. The encryption and decryption algorithm is run over random data (which represents an email message) set using the Javascript library in Chrome browser. The performance of RSA algorithm is studied here in a browser environment. The following are the results using a 512 bit key (Figure 8).

It can be seen that at a relatively larger message size, around 212 KB, the time taken for encryption and decryption

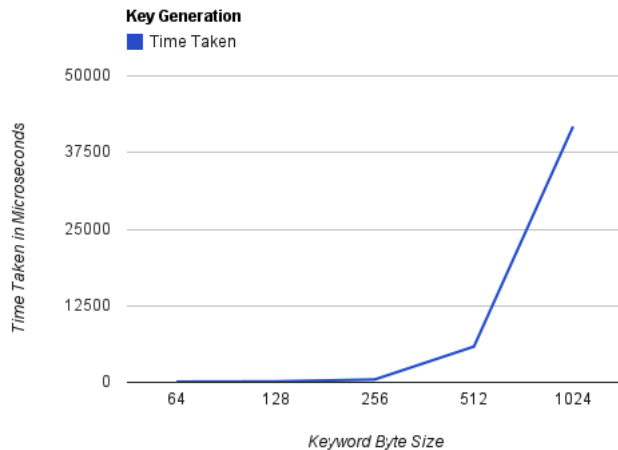


Figure 7. Key Generation

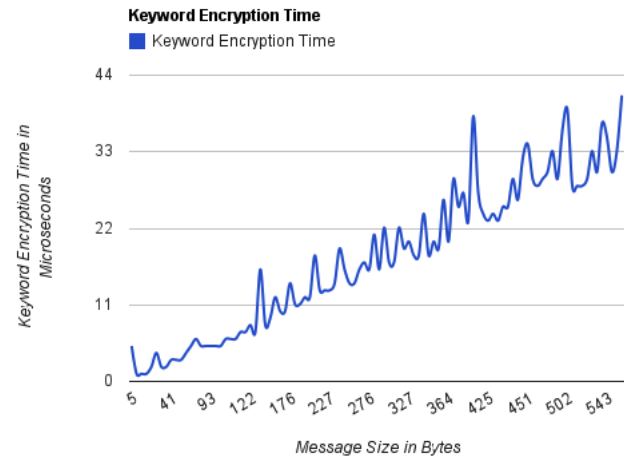


Figure 9. Keyword Encryption Time

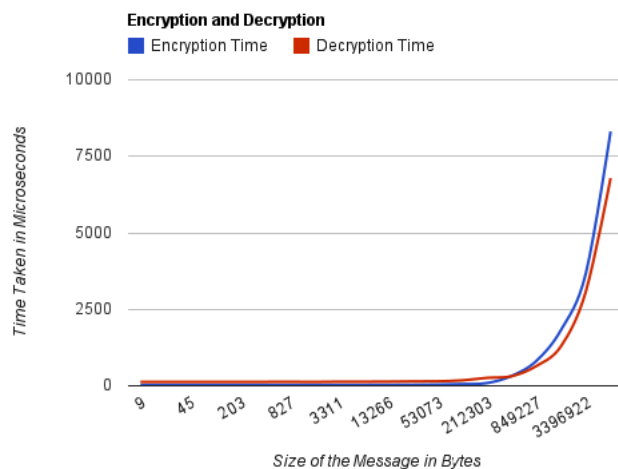


Figure 8. Encryption and Decryption

is less than 2 seconds. However as the message size increases in the order of megabytes, the time is around 16 seconds. A 67 MB message takes around 16 seconds to encrypt and 9 seconds to decrypt, which is still acceptable for sending such a large message. Moreover, most webmail systems have a limit of 10 MB on message sizes.

3) *Keyword Encryption*: In this phase the performance of AES algorithm is studied (Figure 9). Each word from the message is extracted and is encrypted using the AES algorithm. There is no decryption phase here, as the encrypted words are checked against each other.

It can be seen that there is a linear relationship between the message size and time taken for encrypting keywords. It has to be also noted that when there are duplicate words the encryption is not done twice. However, in these experiments each word was generated at random with a random size (with maximum as 25 bytes). The probability of the same word

repeating is very low for this case.

### B. Space Complexity

In our study of the space complexity, we were interested in the following:

- 1) Increase in size of the keyword index
- 2) Increase in the size of the final mail

1) *Impact of increase in size on the keyword index*: The AES algorithm is executed over the generated keywords and the impact of the size of the encrypted keywords on execution time is examined (Figure 10). There is close to a 10 times increase in the generated encrypted keywords compared to the keyword's actual size. This can pose a design challenge at the database level on how to store these keywords for efficient lookups at the server level.

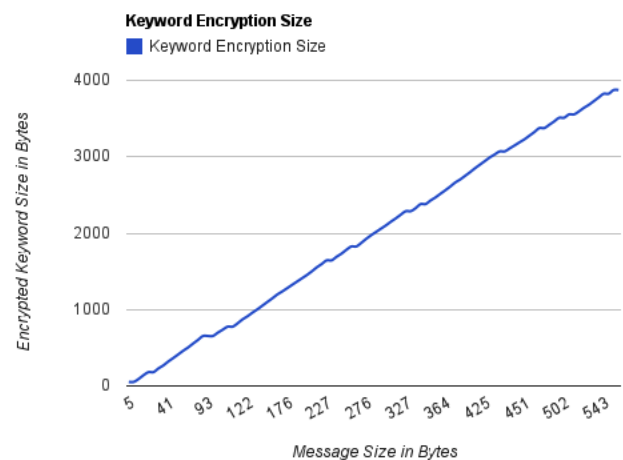


Figure 10. Keyword Encryption Size

2) *Impact of increase on Final Message size:* Here we study the total increase in the email size. The email that is sent to the server of the recipient will be in this format and the any increase in size, will increase the overall network traffic.

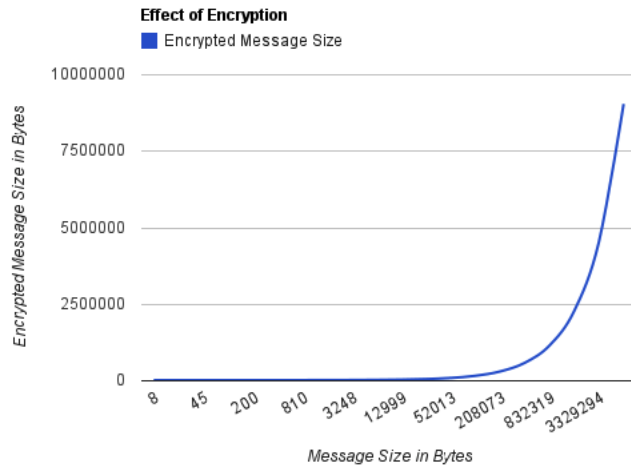


Figure 11. Message Size

It can be seen from the graph (Figure 11) that initially, when the message is transferred, there is not much of an increase in the encrypted message size (8 bytes to 186 bytes, 18 bytes to 199 bytes, 404 bytes to 722 bytes). However as the size increases beyond 4MB there is a steep increase in the difference between the message size and encrypted message (4MB to 5MB, 8MB to 11MB, 66MB to 90MB). On average, there is a 3 times increase in size when encrypted using RSA. This is another major factor that has to be taken into consideration while using this system.

## VIII. CONCLUSION

We proposed a privacy preserving architecture for our webmail system, that enables secure communication of messages using a public/private key model and privacy preserving keyword search functionality using AES key encryption algorithm.

Our approach requires every client to install an extension to their browser and the cloud computing provider to modify their webmail application to support encrypted keyword search. Even though technically this is a possible solution, economically a cloud provider might not prefer this approach. Most of the business models in web application are built around the contextual advertising model, where the cloud provider relies on the user's data to deliver the relevant advertisements to the user. In our case as the data is encrypted in the server, the cloud provider will not have access to the user's data. Works such as Toubiana et al. [31], try to address this problem by offloading the keyword extraction in contextual advertising to the client

browser. Approaches like [31] needs to be modified for our architecture so that our system remains economically viable.

Unlike in PEKS [25], our system does not use a trapdoor function. This makes our system more susceptible to chosen plaintext attacks. If a recipient of a mail is also a potential attacker, the recipient can eavesdrop the encrypted keyword information sent from the sender to the server, and make a guess on what keyword represents the encrypted cipher by analyzing a number of mails sent to the recipient (attacker) from the same sender. However, our contribution is the proposal of the framework. The encryption algorithms used can be modified to utilize more secure alternatives in our architecture.

Our system makes an assumption that the browser and the browser extension framework is trustworthy. We believe it is a fair assumption, as the user can control and monitor the browser activity and any aberration of browser functionality can be detected by the user (at-least theoretically).

In our performance study, we see a considerable increase in the size of the message and the keywords after encryption. This will have a direct effect in the database storage and the keyword look up time.

We have also not implemented the functionality to add the incoming messages to the encrypted search database. Future work should address this. Future work also involves detailed study on the strength of the encryption, support to range and similarity searches, improvements to the algorithms used whilst maintaining performance.

## ACKNOWLEDGEMENTS

The authors would like to thank the IBM Center of Advanced Studies and NSERC for their funding.

## REFERENCES

- [1] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," *National Institute of Standards and Technology, Information Technology Laboratory*, vol. Version 15, 10-7-09, p. 2, 2009.
- [2] D. Kenny and J. Marshall, "Contextual marketing—the real business of the Internet.," *Harvard Business Review*, vol. 78, no. 6, p. 119, 2000.
- [3] "MarketsAndMarkets.com Cloud computing market - global forecast (2010 -2015)."
- [4] R. Gellman, "Privacy in the clouds: Risks to privacy and confidentiality from cloud computing," in *World Privacy Forum*, pp. 1–26, 2009.
- [5] S. Warren and L. Brandeis, "The right to privacy," *Harvard Law Review*, pp. 193–220, 1890.
- [6] "The United Nations Declaration of Human Rights," *The American Journal of International Law*, vol. 43, no. 2, pp. 316–323, 1949.
- [7] "Top Threats to Cloud Computing V1.0," tech. rep.

- [8] T. Garfinkel and M. Rosenblum, "When virtual is harder than real: Security challenges in virtual machine based computing environments," in *Proceedings of the 10th conference on Hot Topics in Operating Systems-Volume 10*, p. 20, USENIX Association, 2005.
- [9] K. Ramachandran, H. Lutfiyya, and M. Perry, "Chaavi: A Privacy Preserving architecture for Webmail Systems," in *CLOUD COMPUTING 2011, The Second International Conference on Cloud Computing, GRIDs, and Virtualization*, pp. 133–140, 2011.
- [10] J. Postel, "RFC821: Simple mail transfer protocol," tech. rep., 1982.
- [11] T. Dierks, "The transport layer security (TLS) protocol version 1.2," 2008.
- [12] P. Zimmermann, *The official PGP user's guide*. MIT Press, May 1995.
- [13] S. Goldwasser and S. Micali, "Probabilistic encryption {& how to play mental poker keeping secret all partial information," in *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, (New York, NY, USA), pp. 365–377, ACM, 1982.
- [14] C. Gentry, *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.
- [15] A. Yao, "Protocols for secure computations," in *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pp. 160–164, Citeseer, 1982.
- [16] W. Itani, A. Kayssi, and A. Chehab, "Privacy as a Service: Privacy-Aware Data Storage and Processing in Cloud Computing Architectures," in *2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing*, pp. 711–716, IEEE, 2009.
- [17] TCG, "Trusted Computing Group (TCG) and the TPM 1.2 Specification," in *Trusted Computing Group*, 2005.
- [18] "Trusted Computing: An Already Deployed, Cost-Effective, ISO Standard, Highly Secure Solution for Improving Cybersecurity," tech. rep.
- [19] J. M. McCune, B. J. Parno, A. Perrig, M. K. Reiter, H. Isozaki, J. M. McCune, B. J. Parno, A. Perrig, M. K. Reiter, and H. Isozaki, *Flicker: an execution infrastructure for tcb minimization*, vol. 42 of *an execution infrastructure for tcb minimization*. New York, New York, USA: ACM, Apr. 2008.
- [20] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Advances in Cryptology-Eurocrypt 2004*, pp. 506–522, Springer, 2004.
- [21] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-Preserving Multi-keyword Ranked Search over Encrypted Cloud Data," in *IEEE INFOCOM*, 2011.
- [22] E. Kangas and L. President, "The Case for Email Security," *Published as a Lux Scientiae Article*, available at <http://luxsci.com/extranet/articles/email-security.html> (accessed 1 May 2007), 2004.
- [23] L. Kaufman, "Data Security in the World of Cloud Computing," *Ieee Security And Privacy*, vol. 7, no. 4, pp. 61–64, 2009.
- [24] "<http://getfirepgp.org/s/home> (Last accessed on June 23rd 2012)."
- [25] J. Baek, R. Safavi-Naini, and W. Susilo, "Public key encryption with keyword search revisited," *Computational Science and Its Applications-ICCSA 2008*, pp. 1249–1259, 2008.
- [26] "<http://www.hanewin.net/encrypt/rsa/rsa.htm> (Last accessed on June 23rd 2012)."
- [27] "<http://www.hanewin.net/encrypt/aes/aes.htm> (Last accessed on June 23rd 2012)."
- [28] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.
- [29] H. B. Westlund, "NIST reports measurable success of Advanced Encryption Standard - News Briefs - National Institute of Standards and Technology - Brief Article," *Journal of Research of the National Institute of Standards and Technology*, 2002.
- [30] B. Schneier, "Snake Oil. Crypto-Gram Newsletter (<http://www.schneier.com/crypto-gram-9902.html#snakeoil>) [Online on 05th September 2011]," 1999.
- [31] V. Toubiana, A. Narayanan, D. Boneh, H. Nissenbaum, and S. Barocas, "Adnostic: Privacy preserving targeted advertising," in *17th Annual Network {& Distributed System Security Symposium*, San Diego, CA, USA, Citeseer, 2010.

# Organizing Security Patterns Related to Security and Pattern Recognition Requirements

Michaela Bunke, Rainer Koschke, and Karsten Sohr  
 Center for Computing Technologies (TZI),  
 Universität Bremen, Germany  
 {mbunke|koschke|sohr}@tzi.de

**Abstract**—Software security is an emerging area in software development. More and more vulnerabilities are published and highlight the endangerment of systems. Hence, software designers and programmers are increasingly faced with the need to apply security solutions to software systems. Security patterns are best practices to handle recurring security problems. The abundance of documented security patterns calls for meaningful classifications to ease searching and assessing the right pattern for a security problem at hand. Existing classifications for security patterns consider only a small number of patterns and their purpose is often focused on implementation issues. Therefore, we identify missing aspects in existing classifications and the similarities between design and security pattern classifications. Based on that, we introduce two new classification schemes. The first is based on application domains formed by a literature survey on security patterns published in the period of 1997 to mid-2012 to cover the whole bandwidth of existing security patterns. The second is based on a subset of the collected patterns that are concerned with software and combines pattern-recognition needs and security aspects.

**Keywords**—Security Patterns, Design Patterns.

## I. INTRODUCTION

Existing security pattern classifications are often based on a few security patterns. Their scope is often limited to special areas such as implementation patterns. In addition, the heterogeneity of the published patterns in this context is very high. In that context, our paper provides a systematic literature review of published security patterns in the period of 1997 to mid-2012. We propose two new classification schemes. The first summarizes all collected security patterns and organizes them into application domains. The second shows in detail which security and implementation forces security patterns with respect to software have. This classification is an extension of our previous work presented at the International Conferences on Pervasive Patterns and Applications (PATTERNS 2011)[1]. We updated our previous work with new security patterns published till mid 2012, enhanced it with a comparison of design and security-pattern classifications and depict challenges in organizing security patterns.

In the domain of software development, **design patterns** have been proposed as specific solutions for recurring

problems in software design [2]. These patterns are also often called **software-design patterns**. Yoder and Barcalow summarized some existing patterns targeting security and introduced the term *security pattern* [3], only three years after Gamma et al. [2] proposed their design patterns. **Security patterns** are best practices aiming at ensuring security [4], [5]. Later on we will use the terminology **software-security patterns** that describe software-related security patterns. These patterns describe security aspects relevant in software design, development, and maintenance.

Existing pattern classifications are mostly based on a small subset of patterns. Their scope is often limited to special areas such as implementation patterns. For instance, Hafiz et al. formed their classification with only 14 security patterns [6], but there exist many more security patterns. Another problem, however, is that information-security experts are rarely development experts [7]. Thus, the usage of existing security patterns and selecting them by way of a classification is a difficult task for non-security professionals who are interested in security aspects.

Therefore, we conducted a systematic literature review and collected the published security patterns in the period of 1997 to mid-2012. We propose a new classification scheme that summarizes 415 security patterns, a much longer list than we found in three surveys [8], [9], [10] and the one by Yoder and Barcalow [3]. Moreover, we shaped this classification scheme towards the selection by application domains, which is relevant for researchers and practitioners who are interested in security patterns.

Retrofitting security aspects into a software system is a difficult task [3]. Accordingly, it would be useful to know which security aspects are already implemented in a software. For instance, Gamma's design patterns can be detected automatically in software systems, but as far as we know, no such approach exists for security patterns [11]. Design and security patterns seem to be very similar except for the security factor, but their concrete similarities and differences are still an open issue in research [11].

Hence, we inspect design and security classifications to determine their similarities and differences to derive possible criteria for a classification that reflect pattern recognition requirements. We use the software-security patterns of our

application-domain classification as a base for this further distinction. Furthermore, our classification is inspired by pattern-recognition needs and combined with the security issues that these patterns solve. For example, this classification can be used by software developers to choose patterns according to particular security requirements. We will use this as a basis for our further research in security-pattern recognition and validation.

The remainder of this paper is structured as follows. An overview of classifications in general is given in Section II. Existing classification approaches for security patterns will be described in Section III. In Section IV, we describe our literature survey and depict challenges in categorizing security patterns in Section V. Afterwards, we will introduce our application-domain classification and specific classification for software-security patterns in Section VI and VII. In Section VIII we will discuss the two presented classifications. Finally, we will conclude and give an outlook in Section IX.

## II. REQUIREMENTS FOR CLASSIFICATIONS

The increasing number of patterns makes it necessary to develop classifications. This section describes requirements for classifications in general and on security patterns in particular.

A classification should be based on systematic methods and techniques to organize a mass of patterns. A classification organizes patterns into groups of patterns that share one or many properties such as the application domain or a particular purpose. The kind of properties that should be used is not fixed and can be customized according to one's needs. A pattern can have more than one specific property. Therefore, it may be included in more than one classification category.

According to Buschmann et al., a pattern classification scheme should meet some basic properties [12]. It must both be simple and easy to learn. This should be supported by using only a few classification criteria to reduce the complexity for users. In addition, a classification should reflect the main properties of a pattern to classify. Last but not least, a classification scheme should provide the possibility to classify new patterns.

Fernandez et al. pointed out that a classification should make the application of patterns much easier along the software life-cycle [13]. Because it is impractical to look at all details of all patterns during pattern selection for the problem at hand, a classification should help to understand the essential nature and value of patterns.

A natural way to classify patterns is to categorize them according to the criteria shown in Figure 1. A simple and intuitive classification can provide one or more of these criteria:

- *Discipline* - categorize patterns according to the discipline when they are applied such as requirements or

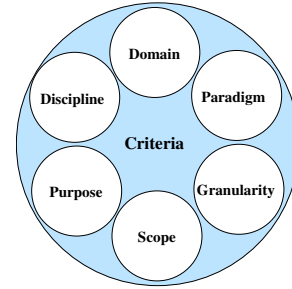


Figure 1. Intuitive classification.

reverse engineering.

- *Domain* - differentiate patterns by their application domain such as network, embedded systems, or distributed systems.
- *Granularity* - rank patterns depending on the level at which they address a system, e.g., they may address software design or coding patterns.
- *Paradigm* - sort patterns according to paradigms, e.g., programming paradigms such as object-oriented or imperative programming.
- *Purpose* - order patterns by the kind of problem a pattern solves and the point in time it may be applied.
- *Scope* - organize patterns with regard to the characteristic of using them, e.g., class or object representation (see [2]).

## III. EXISTING CLASSIFICATIONS

The presented classification criteria in Section II are simple, but do not always fit for selecting the right pattern for a special purpose because of their generality. Therefore, more specific classification schemata based on one or more criteria have been developed to meet special purposes. Due to the fact that security patterns are formed according to the archetype of design patterns, we will start with classifications for design patterns and continue with existing security-pattern classifications. We will close this section by discussing gaps in security-pattern classifications and whether design-pattern classifications can be used to classify security patterns.

### A. Design-Pattern Classifications

Gamma et al. introduced the first classification of design patterns (GoF patterns) [2]. GoF (Gang-of-Four) patterns is an alternative name for the design patterns introduced by Gamma. They classified their patterns based on two criteria: scope and purpose.

As depicted in Figure 2, the “scope” dimension is distinguished by object composition and class inheritance. The purpose dimension is split into *creational*, a *structural* and a *behavioral* criteria. A pattern that is related to an object creation fits into the *creational* criteria. If a pattern is concerned



Scope \ Purpose	Creational	Structural	Behavioral
Class Related	Factory	Adapter (class based)	Interpreter
Object Related	Singleton	Facade	Iterator

Figure 2. GoF classification with a few examples [2].

with compositions or structures that are created by classes or objects, it is called *structural*. The last criterion *behavioral* deals with the way communication or responsibilities are distributed.

Zimmer organized the GoF patterns according to their relationships [14]. He classified the relationships in pairs  $(X, Y)$  where  $X$  and  $Y$  are different design patterns. The relationships are defined as follows:

- $X$  uses  $Y$  in its solution,
- $X$  is similar to  $Y$ ,
- $X$  can be combined with  $Y$ ,

With these categories, he introduces a new layer structure for pattern classification. According to Figure 3 relationships and structure of patterns are distinguished into three layers:

- Basic design patterns and techniques
- Design patterns for typical software problems
- Design patterns specific to an application domain

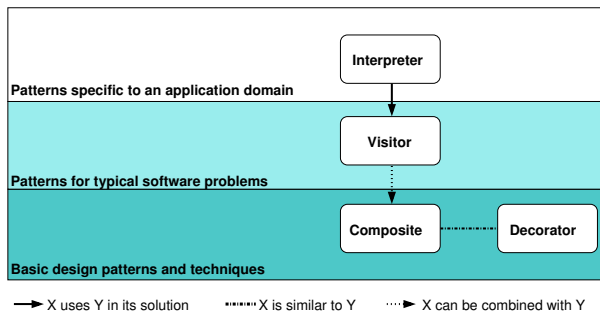


Figure 3. Zimmer's classification with a few examples [14].

Later on, Buschmann et al. presented another organizing approach [12]. They state that all patterns reside on different abstraction layers and it would be more useful to organize them into criteria that express their abstraction level. Therefore, the authors divide their own patterns into three kinds of patterns:

- *Architectural patterns*: specify the fundamental structure of applications.
- *Design patterns*: describe often occurring structures of software-component communication that solve a recurring design problem for a specific context.
- *Idioms*: coding patterns, that is, proven conventions and techniques used during the implementation phase of an

application.

Some patterns depend on the technology or domain they are used for and implemented in. These are so-called domain-dependent patterns, e.g., Java Platform, Enterprise Edition (JEE) design patterns. These patterns can be used only in the JEE environment. A system of such patterns has been described by Alur et al. [15]. The authors want to keep the classification simple for their patterns, so they assume “each pattern hovers somewhere between a design pattern and an architectural pattern”. These patterns can be classified in the following categories according to their logical tiers (see Figure 4). The *presentation tier* is responsible for creating the presentation used by the client to interact with the user. The *business tier* is responsible for executing the business logic of the application and applies the business logic to the information received from the integration tier. The *integration tier* performs the data-access operations for the application.

Patterns	Tier
Composite View	Presentation
Service Locator	Business
Service Activator	Integration

Figure 4. JEE pattern classification with a few examples.

The design patterns we are discussing exist since 1994. Classifying them is not a highly active research topic in the design-pattern community. An exception is arising new technologies like JEE which require new classifications or the re-evaluation of existing ones. The older ones were not refined further, except for some theoretical abstractions like the one by Hasso and Carlson [16]. They use a complex algebraic structure to classify design patterns.

In 2006, Shi and Olsson identified a lack of classifications for the need of design-pattern recognition [17]. They decided with the hidden agenda of detecting design patterns a new classification approach for the 23 GoF patterns. They suggest a reclassification related to the need of pattern detection by using five categories *language provided*, *structure driven*, *behavior driven*, *domain specific* and *generic concepts*. When a pattern is implemented in some programming language and can be identified by looking at the inheritance hierarchy or specific method names, the pattern is part of the *language provided* category. Patterns that are deeply shaped by their structure and can be identified by their inner-class relationships such as the *Bridge* or *Composite* pattern are *structure driven* patterns. Patterns that have a structure coupled with a specific behavior fit in the category *behavior driven* such as *Singleton* or *State* pattern. Patterns such as *Interpreter*



or *Command* serve domain-specific needs. Detecting such patterns requires domain-specific knowledge. They belong to the *domain specific* category. Patterns in the category *generic concepts* lack a definite structure and behavioral aspects such as the *Memento* pattern.

Their classification allows them to exclude the domain-specific patterns and generic concepts, which cannot be found with common behavioral and structural pattern detections. Moreover, they excluded the language-provided patterns from their detection process because of their easy detection using name matching, too. Patterns that reside in the categories *behavior driven* and *structure driven* were used for design-pattern detection in their tool PINOT. After Shi and Olsson's classification approach, no new design pattern schemata have been developed to the best of our knowledge.

### B. Security Pattern Classifications

One of the simplest classifications for security patterns was used by Kienzle et al. [18]. They presented the *structural* and *procedural* criteria for the differentiation of the patterns described in their final report. If a security pattern is concerned with compositions or structures that are implemented in a software product, it is *structural*. If a security pattern improves the process for developing secure software with regard to the organization or management, it is called *procedural*.

Konrad et al. [19] proposed a classification method for security patterns by re-using the classification for design patterns such as *creational*, *structural* and *behavioral* from Gamma et al. [2]. They enhanced their classification by adding further categories such as network, host, and application (see Figure 5). In their work, they considered only the security patterns introduced by Yoder and Barcalow [3].

Purpose Abstraction Level			
	Creational	Structural	Behavioral
Application	Session	Check Point Authorization	Limited View Full View with Errors
Host	Session	Check Point Authorization	_____
Network	Session	Check Point Authorization	_____

Figure 5. The classification by Konrad et al. with a few examples [19].

Schumacher's security patterns book offers a new classification system [20]. The classification is based on Zachman's framework for enterprise architecture [21]. It is presented along two dimensions. One dimension represents different views on the interrogatives "what", "how", "where", "who", "when", and "why". The second dimension shows different information model views such as *business model* or *technology model*. Schumacher et al. enhanced this framework by adding the column *security* to emphasize the security

view and to be able to address all model levels. They organized only the patterns contained in the book into their classification.

According to the JEE pattern classification by Alur et al. (see Section III-A), Steel et al. classify their JEE security patterns in a similar way [22]. They separate their patterns in layers that are typical for the development in the JEE domain such as *Web*, *Business*, and *Web Service*, and added a fourth tier that represents the special issue of *identity management* (see Figure 6). This classification is designed only for the special purpose of JEE patterns and does not consider other types of patterns.

Patterns	Tier
Authorization Enforcer	Web
Secure Session Object	Business
Secure Message Router	Web Service
Password Synchronizer	Identity

Figure 6. The classification by Steel et al. with a few examples [22].

Rosado et al. related security requirements to security patterns and classified security patterns into two categories: architectural and design patterns [23].

Hafiz et al. note that simple security-classification concepts are not sufficient to create a partition of security patterns [6]. Their focus is to classify security patterns by their security impact. Their subset of 14 different security patterns is organized by a classification of application context, a Microsoft classification scheme, the CIA [24] and STRIDE model [25]. The acronym STRIDE contains the concepts **S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service, and **E**levation of privilege. Moreover, they proposed a classification based on a tree structure combined with the STRIDE model to join the software and security view in terms of security patterns [6]. The STRIDE model is normally used for threat modeling including identification and prioritization of security vulnerabilities. It is a common tool for security architects who have to prioritize the mitigation effort of security techniques.

VanHilst et al. introduced a multi-dimensional matrix of concerns to classify security patterns [26]. It addresses the problem coverage and pattern classification. Their idea was that each matrix dimension represents a well-defined list of concerns. To classify security patterns, the primary dimension contains concerns of life-cycle activities, such as *domain analysis* or *requirements engineering*. The second dimension differentiates security patterns by their component source type such as *new code*, *legacy*, or *wizard-code*.

Other dimensions may hold types of security responses like prevention or mitigation, but they can also be further customized to a user's need. Their classification was tested by different members of their team, who added six different security patterns to the classification.

Fernandez et al. state that security patterns are architectural patterns [27]. On that account, their approach deals with two classifications that differ in different viewpoints of security patterns. On the one hand, they introduced a classification by a hierarchy of layers and on the other hand, they proposed a classification based on the relationships between patterns by using an automatic relationship extraction and analysis technique. This classification is abstract and regards only a small number of security patterns.

Washizaki et al. point out that the previously introduced classifications have only a few dimensions and do not embrace the relations between patterns [28]. Hence, they introduce a meta model to express the patterns' properties and relations uniformly. The base is an excerpt of the multidimensional classification dimensions presented by VanHilst et al. [26]. They selected the dimensions as follows: *Lifecycle stage*, *Architectural level*, *Concern*, *Domain*, *Type of pattern* and *Constraint*. In addition, they used the three UML standard relationship types *association*, *generalization*, and *aggregation* to model relationships between security patterns, for example, the *Firewall* pattern [20] is the generalization of the *Address Filter Firewall* [29] and the *Application Firewall* [30] pattern.

They also propose two instances for the meta model that represent two points of view, namely pattern-to-pattern relations, represented as a pattern graph, and pattern-to-dimension relations modeled as a dimension graph. They tested their approach with only eight different security patterns that are close to implementation patterns.

### C. Classification Similarity

There exists an evolution of design and security pattern classifications with respect to the used classification ideas. We show the influence among security and design-pattern classifications in Figure 7. Some ideas like the purpose of the GoF's design-pattern classification were reused by Konrad et al.'s security pattern classification. Moreover, the criterion *structural* has been adapted by Kienzle et al., whereas the criterion *procedural* and *behavioral* in the GoF classification have different meanings. *Procedural* is used with respect to process patterns for the management or organization of software development in contrast to *behavioral* from the GoF classification where patterns are only software patterns that will be implemented in a software system.

The criteria *architectural* and *design* were proposed by Buschmann's classification scheme et al. [12] and picked up by Rosado et al. [23] and used in conjunction with requirements for a new classification schema.

The three-tier JEE classification [15] and the four-tier

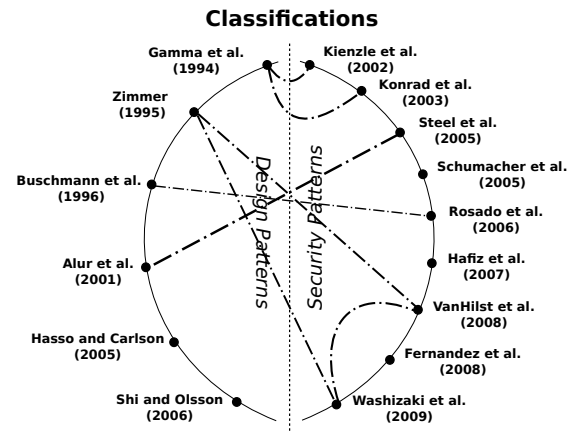


Figure 7. Design and security pattern classification relations – dashed lines between publications highlight alike ideas.

classification for security patterns [22] are very similar, too. They differ only in one additional tier by the security patterns, which deals with identity information. The other three tiers have sometimes different names *Presentation/Web*, *Business/Business* and *Integration/Web Service*, but describe the same criterion (see Figure 4 and 6).

Organizing patterns according to their relationships was introduced by Zimmer and reused by Fernandez et al. and Washizaki et al., but their relations are different. Zimmer [14] depict a graph with only three predefined types of relationships and Washizaki et al. focus on the UML standard to represent pattern relations like generalization or use relationships [28]. In contrast to that approach, Fernandez et al. use automatically extracted relationships based on the pattern description [27].

### D. Classification Distinction

We showed that security-pattern classifications were influenced by design-pattern classifications. All published classifications have one element in common: they take only a small number of patterns into account. On the security-pattern side, the used patterns are often very similar to the patterns first introduced by Yoder and Barcalow [3] and on the design-pattern side the approaches often consider the “core” design patterns described by Gamma et al. [2]. Both subsets of patterns are patterns that will be implemented in a software system. This may lead to the impression that only a handful security and design pattern exist and imply that only programming issues are covered by these patterns. However, Henninger et al. [31] showed in 2007 that there exist more than the few patterns published by Gamma et al. [2] and Buschmann et al. [12]. This statement can also be extended to security patterns, which can also describe enterprise or other security related issues.

Most security pattern classifications are more complex to cover more properties or split purposes or domains into more

dimensions than design patterns. Because of that, we can assume that the security pattern community is aware of the security pattern's heterogeneity, which reflects the additional dimensions in the security-pattern classifications. Design-pattern classifications are often tailored to one group of interest – the software developers. Hence, they are focused on helping to choose the right pattern in design and developing to reach a good code quality and system structure. Due to the fact that the security-pattern audience has more than one group of interest such as security, software development, or enterprise-process design, the security-pattern classifications are built from more heterogeneous criteria than the design-pattern classifications. Yet, not all interests can be covered in one classification. Therefore, more security classifications have been developed till now and developing new ones is still a current topic in the security-pattern community.

A security view is often added to these classifications by using common-threat modeling such as CIA [24] or STRIDE [25]. Adding new views increases the complexity of a classification. A problem, however, is that information security experts are rarely development experts [7]. Because increasing the complexity in security-pattern classifications can make the usage of a classification more difficult for users that have no knowledge or experience with security. It may also lead to difficulties in understanding for other interest groups within the addressed security-pattern audience.

#### E. Summary

Security patterns related to software can be categorized in a way similar to design patterns. Security patterns that describe other aspects than software-related issues cannot be distinguished by the criteria the aforementioned design-pattern classifications offer, such as “tier”, “class related” or “language provided”. Therefore, we plan to classify the security patterns in two steps. First, we will look at all security patterns and organize them according to their application domain with respect to their heterogeneity. Secondly, we will focus on software-related patterns and will develop a new classification with existing criteria of design-pattern classifications with respect to software-security patterns.

### IV. COLLECTING SECURITY PATTERNS

Existing security classifications are limited by the number of chosen security patterns. In addition, existing security pattern surveys are biased by their focus on the same set of security patterns such as the ones of Yoder and Barcalow [3]. The SecurityPatterns website, which provides a short list of security patterns, offers a few more patterns, too, but mixed with articles that describe the application of security patterns [32]. This was not an appropriate position for starting our research activities. Therefore, we decided to conduct a literature survey to provide a proper background for our new classification approach, which should cover the whole range of published security patterns till today. This

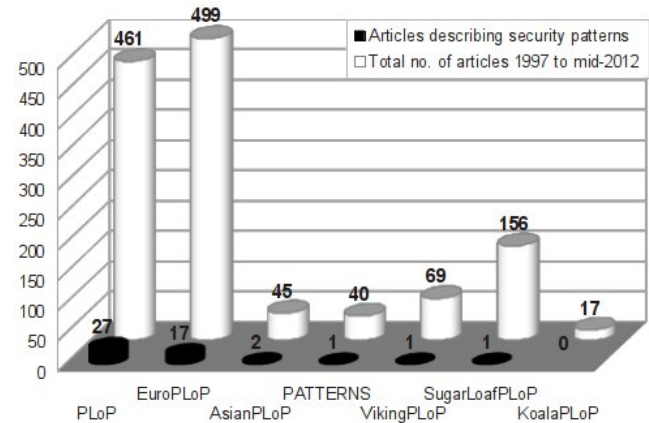


Figure 8. Conferences Involved in the Initial Article Selection.

section describes how our literature survey was conducted systematically following the guidelines by Kitchenham [33], [34].

#### A. Article Selection and Discovery Process

We started our literature research with the surveys carried out by Laverdiere et al. [8], Heyman et al. [9], and Yoshioka et al. [10]. Their surveys give a good overview of published patterns. Hence, we also considered common pattern-related conferences (see Table I). and looked for security patterns in the IEEE Digital Library [35] and the ACM Digital Library [36] and considered two security patterns books [20], [22].

The discovery process was split into two parts. One part is the selection of articles published at pattern conferences and the second part is the search for electronic publications.

*1) Searching Through Pattern Conferences:* The aforementioned pattern conferences (see Table I) of the years 1997 to mid-2012 were skimmed for several keywords, such as cryptographic, security, software, or secure. At first, we picked out all publications that contain these keywords. In this initial selection phase, we found 1268 articles (see Figure 8). Secondly, we read the abstract if it described the presentation of a security pattern and made a note of the authors, publication year, and title. Thereafter, we read the publications not filtered out previously to verify that they describe security patterns. In this step, we enhanced our list with each identified pattern for further readings. Finally, we scanned the publication references and collected referenced publications containing the aforementioned keywords. We stopped the step of cross-reference scanning when we did not find new publications containing the keywords we searched for.

*2) Searching Through Electronic Publications:* On searching for other electronic publications we used the two

Acronym	Description
AsianPloP	Asian Conference on Pattern Languages of Programs
EuroPloP	European Conference on Pattern Languages of Programs
KoalaPloP	Australian Conference on Pattern Languages of Programs
PATTERNS	International Conferences on Pervasive Patterns and Applications
PloP	Conference on Pattern Languages of Programs
SugarLoafPloP	American Conference on Pattern Languages of Programming
VikingPloP	The Nordic Conference on Pattern Languages of Programs
GRID-STP	International Workshop on Security, Trust and Privacy in Grid Systems
ICOMP	International Conference on Internet Computing
IFIP WC 11.3	Working Conference on Data and Applications Security
SECURWARE	International Conference on Emerging Security Information, Systems and Technologies

Table I  
CONFERENCES INVOLVED IN THE INITIAL ARTICLE SELECTION. A GREY BACKGROUND INDICATES THE DISCOVERY BY CROSS-REFERENCES.

digital libraries provided by IEEE and ACM [35], [36]. Both offer an extensive database search for published publications. First of all, we used the simple search to find publications that contain the aforementioned keywords. Due to the fact that the number of results was very high and too unspecific, we used the advanced search field provided by the websites to obtain more localized results. There, we concatenate the following options with "and" and limited them to get better results:

- The year of publication date has been limited from the year of the first published security patterns 1997 to mid-2012.
- The full text must contain the word "pattern".
- The title must contain one of the aforementioned keywords.

Unfortunately, these restrictions still provided many unwanted results. Therefore, we skimmed the result list from top to bottom – where the search engines of the digital libraries provided the ordering based on relevance – and discontinued if we read more than ten papers that do not deal with security patterns in their abstract.

Further, we skimmed the collected patterns like the aforementioned conference publications. We verified that they describe one or more security patterns and then collected their cross references.

### B. Summary

We identified 67 different publications describing security patterns, including books, journals, proceedings, and technical reports. Most of them were found by looking at the Hillside Group [37] pattern conferences such as PLoP and EuroPloP (see Figure 9 and Table I) and books. Another publication type containing many security patterns were technical reports discovered by cross references. New conferences that have only a few security-pattern publications are also discovered by cross references (see Table I).

The search at the ACM and IEEE Digital Library produced many false-positive articles that were at a closer look no security pattern descriptions, but deal with them in other

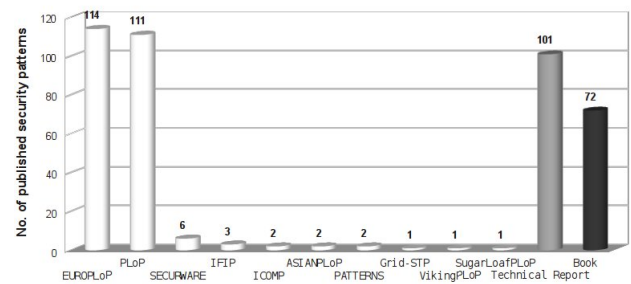


Figure 9. Distribution of security patterns across different venues; white bars denote conferences, the grey bar technical reports, and the black bar books.

ways like discussing secure software design in practice [38].

Some publications describe more than one pattern. In total, we got 415 security patterns. This list identified that some of these patterns have been described more than once. Hence, we filtered out duplicates and reduced the number of patterns to 364. These duplicates were identified by the use of similar names and then comparing their descriptions. Because of the abundance of patterns, we were not able to check in depth whether two patterns with different names relate to the same concept. This was also forced by the nonuniform descriptions of the patterns, which will be discussed in the next section.

## V. CHALLENGES IN CATEGORIZING SECURITY PATTERNS

Identifying duplicates is not the only challenge in categorizing security patterns. We agree with Yoshioka et al. [10] and Heyman et al. [9] that the abstract description of patterns is another challenge. Because the quality of the security-patterns descriptions may influence the categorizing outcome, we spent some time in inspecting the description forms.

### A. Description Form Inspection

Various descriptive models for security patterns exist within the security-pattern community [39]. The POSA<sup>1</sup> model described by Buschmann et al. is said to be frequently used to describe the context and usage of security patterns [12]. Yet, during our literature review we observed that the patterns are often described in custom styles and do not follow strictly the POSA model.

Therefore, we gather all used description aspects in the collected security-pattern publications to review how security patterns are described formally. We assume that description aspects that are often used in several pattern descriptions provide the best clues for selecting and organizing security patterns.

### B. Section Assessment/Examination

There exist 63 different sections or aspects such as *Problem*, *Intent* or *Known Use* that are used in the 67 collected security pattern publications. The heterogeneity in naming the aspects is very high and no mapping between the different names exists like the one by Henninger et al. between the POSA and GoF descriptions [31]. For this reason, we identified significant sections which can be used to get a first impression on security patterns and make patterns comparable (see Table II).

**Context** is a frequently used description aspect with 49 of 67 hits, but the context description is often very short. In some publications, it consists only of one or two sentences (see [40] or [41]). This circumstance makes it hard to obtain sufficient knowledge or even an idea of what the pattern is about. Similar findings were made by Laverdiere et al. for the naming and the section *Intent* in security-pattern descriptions [8].

The **Problem** aspect occurs in about 84 percent of the publications. In many cases, these problem descriptions are abstract or describe a simplified problem for the security pattern (e.g., [40]). Therefore, this description aspect is less applicable to categorizing security patterns for an application domain but suitable to gather the security aspects it addresses.

The **Related Patterns** aspect requires a good knowledge of other security patterns and their application domains to be used for a distinction. This also applies to the **Consequences** aspect where additional knowledge is required to be able to relate to application consequences in the security area. Hence, these sections cannot be recommended for novices in security to accomplish a pattern distinction.

A **Known Use** aspect depicts where a pattern can be found in real life. This section often labels software or software parts like an application login screen, UNIX telnet or Linux as operating system software. The given keywords

Sections	Used by # Publications
Solution	58
Problem	56
Related Patterns	50
Consequences	50
Context	49
Known Use	46
Example	33
Forces	27
Example Resolved	25
Structure	25
Implementation	23
Dynamics	21
Intent	22
Also Known As	10
Motivation	10
Participants	8
Applicability	7
Variants	7
See also	7
Collaboration	5
Sample Code	5
Alias	4
Impairments	3
Resulting Context	3
Abstract	3
Benefits	2
Features	2
Properties	2
Preconditions	2
Resultant Context	2
Running Example	2
Alternatives	1
Class-Diagram	1
Classification	1
Conflicts	1
Contradictions	1
Dependencies	1
Design Issues	1
Example Instances	1
Hardware/Software	1
Implementation Factors	1
Implementation Issues	1
Implementation Example	1
Issues	1
Labels	1
Liabilities	1
Non-Security Known-Use	1
Non Software Example	1
Other Example	1
Participants & Responsibilities	1
Rationale	1
Reality Checks	1
Relationships	1
Resolved Example	1
Resulting Context	1
Security Factors and Risks	1
Security Objectives	1
Solution Example	1
Solution Implementation	1
Social Dependencies	1
Specific Context	1
Strategies	1
Trade-Offs	1

Table II  
ASPECTS WHICH ARE USED BY SECURITY-PATTERN DESCRIPTION FORMS. THE OFTEN USED ASPECTS ARE HIGHLIGHTED WITH A GREY BACKGROUND.

<sup>1</sup>POSA is the acronym of the design-pattern book series "Pattern-Oriented Software Architecture" written by Buschmann et al. [12].

and explanations within this section give a good impression to which domain this pattern can be applied.

A **Solution** aspect is used in about 87 percent of the 67 publications. The described solution in the collected publications is frequently used and provides often a good depiction of the security pattern and the problem it solves. Moreover, this section often describes which security aspects are covered by the pattern and how this could be implemented by software-security patterns. If the description does not provide a solution aspect, it can hardly be considered a pattern description. One may argue that the GoF and POSA description templates neither provide a solution aspect, but they describe this aspect in a refined manner using the description aspects *Structure*, *Implementation* and *Running Example*. This heterogeneity in the form of descriptions is one aspect that we have to deal with and will be discussed in the following section.

### C. Challenges for our Classification Approach

Besides the high variation in the pattern description quality, we note that not all often occurring aspects are equally useful to get a quick access to a pattern's goal and application domain.

If one does not have the time to read a whole pattern description or has a lack of sufficient security knowledge to understand the described pattern, we propose to look at first at the *Known Use* aspect to get an idea of the pattern's application domain. On account of the good depiction of the security pattern and the problem it solves in the *Solution* aspect, we can recommend in a second step to look at this section, if there exists no *Known Use* section or if the containing information is not satisfactory. In addition, the *Solution* aspect gives hints on how the pattern can be implemented in software or used for end users or enterprise processes.

With this strategy, approximately 80 percent of the patterns can be sufficiently understood. The remaining 20 percent can only be organized by reading the full pattern description because of their insufficient description structure in comparison to the majority of security publication. So we decided to read the whole pattern description for each classification because of the high description heterogeneity and high varying description quality.

During the pattern description-form examination process we observed that some description-form aspects are filled in an insufficient way. An example is the publication by Yskout et al. where many aspects in the pattern description form exist, but many of them are filled with one or two words or with a few sentences [42]. Due to the fact that such an imprecise description leaves much room for interpretation and imagination about what the pattern describes, it increases the difficulty in the distinction process for a new classification. It may also compromise the correctness of distinction.

Many security-pattern description forms follow in some

aspects the POSA Template, but they are compounded by different terminology like *Problem* and *Motivation* or *See Also* and *Related Patterns*, which describe the same issue in the publications. A uniform form of description is desirable. Research should aim at improving the quality of security-pattern descriptions. Initial work along this line has been done but only for a small subset of patterns [9], [43].

## VI. APPLICATION DOMAIN CLASSIFICATION

The new classification unifies the existing patterns into a common scheme. In addition, not every task needs information about attack surfaces or vulnerability classification properties like STRIDE or other facets that are introduced in Section III. On that account, we omit specialized criteria like STRIDE and focus on universal differences among the security patterns. With this in mind, we develop a new classification with a more general perspective based on a domain criterion (see Section II) and the security patterns we collected in our systematic literature review (see Section IV).

### A. Organizing by Application Domain

To derive our classification, we first skimmed over the data and collected keywords for the security patterns such as user, password, operating system, enterprise or process. These keywords were gathered by information we found in the pattern descriptions.

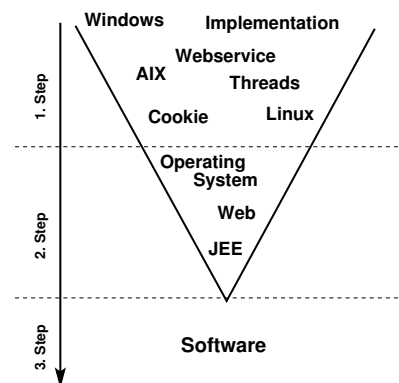


Figure 10. Proceeding steps in our classification model.

In the next iteration, we went through the pattern list and extracted keywords for the patterns. On further reading, these keywords were unified into common groups. For instance, we united the keywords *AIX*, *Linux* and *Preforking* to the group *Operating System*. The result contains a mixture of purpose and domain criterion. We formed 13 different groups this way. To further simplify the classification along the lines described in Section II, these keywords were further condensed to form an application-domain based distinction, which is easy to understand and intuitively applicable (see



Figure 10). Finally, Figure 11 depicts the five target application domains that were discovered: *Enterprise*, *Software*, *Cryptographic*, *User*, and *Network*. They are described in the following in more detail.

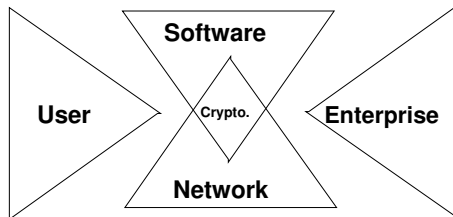


Figure 11. Application-domain based classification.

### B. The Application-Domain Criteria

**Enterprise**-security patterns deal with aspects that are important for enterprises to ensure security in several enterprise segments such as third-party communication with suppliers. This means security in processes, physical authentication to several areas, risk mining or securing communication in internal and external businesses. A good example of this pattern type is the *Manage Risk* pattern introduced by Elsinga and Hofman [44]. The problem addressed by this pattern is as follows: “What is the right (combination of) paradigm(s) to formulate the corporate security strategy in order to select and implement the appropriate set of security safeguards?” The pattern suggests to instruct people and units to pay attention on known and unknown risks to develop prevention and roll-back strategies.

**Network**-security patterns address network infrastructures and their ideal composition. For instance, the *Packet Filter Firewall* pattern describes how to shield an internal network from Internet attacks just by tunneling the communication traffic through a single controllable instance [20] and the *Virtual Private Network* pattern [39] depicts how secure connections over public networks such as the Internet can be established. The point-to-point tunneling protocol (PPTP) is a specific implementation of this pattern [97].

**User**-security patterns are focused on user behavior or their awareness of security issues, for example, the *Password Lock Box* pattern, which encourages the user to protect master passwords with the highest level of security [52]. It stresses the significance of protecting master password files and depicts situations where such a file can be useful. The *Keep It Secret* pattern [52] highlights that published or publicly known passwords pose a potential danger to be misused by attackers. To minimize this effect, one should keep a password secret or use *Password Salt* (another security pattern) to vary the password [52]. Another pattern in this domain describes how one can configure the web browser to control how and when cookies are set and used [54].

**Software**-security patterns describe mostly how to structure parts of software to ensure security requirements. Sometimes they also describe a specific behavior or way to manage or control a data flow in a secure way. On one hand, patterns in this domain can be very specific like JEE patterns, which can be applied only to Java enterprise applications [22]. An example is the *Container Managed Security* pattern [22], which is a standard way to enforce authentication and authorization in a JEE application so that no special hard-coded security policies are necessary. On the other hand, patterns in this domain can be more general like the *Single Access Point* pattern, which models a kind of login structure that can be found in several software systems like UNIX, ICQ or Twitter [3]. Patterns of this application domain can also be called *Security Design Patterns* along the lines of the GoF design patterns, which also focus on software.

**Cryptographic** security patterns depict secure communication between two applications over a network. They are often described abstractly. Therefore, it is not clear whether these patterns reside in the *Network* or *Software* domain. Their implementation or application is possible in both domains. On that account, we view them as a part of network and software in our classification (see Figure 11). An example is the *Sender Authentication* pattern. It presents the problem and solution how to guarantee that a received message has been sent by a person one expected [40]. Obviously, such a pattern can be applied at network level (level 3 and 4) or at application level, and depending on that, it resides on the *Network* or *Software* application domain.

The aforementioned classifications in Section III cover only parts of the fields we discovered. The *Network* domain is partly touched by the classification of Konrad et al. [19]. Schumacher et al. [20] factor *Enterprise* requirements customizable with viewpoints in their classification, but they do not distinguish other domains as our approach does. The domains *User* and *Cryptographic* are not mentioned in the existing classification approaches, although they represent approximately one sixth of the patterns (see Table III).

## VII. MERGING PATTERN RECOGNITION AND SECURITY NEEDS

The application-domain classification scheme can be tailored further to practical or research interests by employing, for example, viewpoints as recommended by Fernandez et al. [27]. For software engineering in particular, applicable patterns are located in the category *Software*, which can be further divided into specific purposes such as pattern detection by using the existing pattern classifications by Shi and Olsson [17]. Developing new viewpoints or finer grained classifications to cover new needs in terms of special purposes for one of the application domains is also conceivable.

An interesting issue for us is the effect of implemented



Application Domain	Publications Describing Security Patterns	Total no. of Security Patterns
Enterprise	[20], [45], [46], [47], [44], [48], [49], [50], [51]	86
User	[52], [53], [54]	24
Cryptographic	[40], [55], [56], [57], [58]	37
Network	[20], [29], [39], [30], [41], [59], [60], [61], [62], [63], [64], [65], [66], [67], [68], [69], [70], [71], [72], [73]	56
Software	[3], [18], [20], [22], [42], [47], [50], [54], [55], [57], [59], [63], [64], [67], [74], [75], [76], [77], [78], [79], [80], [81], [82], [83], [84], [85], [86], [87], [88], [89], [90], [91], [92], [93], [94], [95], [96]	161

Table III  
PUBLICATIONS AND NUMBER OF SECURITY PATTERNS PER APPLICATION DOMAIN

security aspects in software. The usage of security patterns can harden and protect a software against common threats [4], [5]. Halkidis et al. [98] showed that one can determine architectural risks for a software systems through the fact of usage or non-usage of security patterns. Therefore, we assume that software which is designed using security patterns is more secure, and if we are able to detect these patterns, we can rate the degree of security of a software system by the usage of security patterns. This is comparable to software design patterns, which imply a higher code quality when used appropriately.

#### A. Classification

Our approach is motivated by searching for security patterns in software to be able to determine the built-in security mechanisms of a software system. The application-domain classification scheme indicates which security patterns are relevant for designing software. Due to the fact that this classification scheme is very general, we decided to tailor the patterns in the *Software* domain further to our research goal. We chose two dimensions for our classification to show the pattern's security impact and its purpose in terms of software development. The first dimension represents the pattern-recognition aspects and the second common security aspects (see Figure 12). All classified software-security patterns in detail can be found in the Appendix. Our classification dimensions will be described in depth in the following sections.

1) *Pattern-Recognition Aspects*: Early approaches to design-pattern detection date back to the year 1996 [99]. There exist several specialized approaches of pattern recognition that use different aspects of patterns for their detection, such as structural, behavioral aspects or software metrics. The common matching techniques are all based on structural and/or behavioral aspects, e.g., [17], [99], [100], [101]. Patterns of these aspects need different information

and analyses to be automatically detected in a software system. Having a distinction for these aspects is very helpful to define which analyses and what kinds of information are needed to find a specific pattern.

Some existing security-pattern classifications depicted in Section III are formed on ideas from formerly published design-pattern classifications. Thus, they imply that security patterns are like design patterns. We gave another picture of the security-pattern landscape with our application-domain classification. Due to the fact that we extracted software-security patterns of the whole set of security patterns, we can use some of the design-pattern classification criteria for our needs.

Representing information relevant to pattern recognition, we choose some criteria of the design-pattern classification of Shi and Olsson [17]. **Structural** software-security patterns are characterized by their particular class structure. This structure can be realized by inter-class relationships like inheritance, association or delegation relationships (see also [17]). The *Single Access Point* pattern [3], [20] is such a structure-driven pattern. It provides a single access to a protected system. The focus, is on the pattern's structure in a software described through static relations rather than its behavior at runtime. **Behavioral** software-security patterns are primarily designed from a behavioral point of view. They can be easier described and found by their typical behavior than their structure (see also [17]). An example of such a pattern is the *Secure Logger* [22]. It is a class that manages the logging of data in a secure and centralized manner. **Generic Concept** security patterns describe very general solutions for security problems. Unfortunately, many of the patterns in this category do not provide implementation details, UML diagrams or other information, such as example code snippets, that can be used to distinguish between a structural or behavioral character of the pattern. An example is the *Password Authentication* pattern [18], which describes

the secure management of passwords while designing a user login.

Shi and Olsson [17] also provide the criteria *Domain Specific* and *Language/Framework provided* but they cannot be directly used for our security-pattern classification. Due to the fact that security patterns are mostly written in an abstract way it is not possible to classify a pattern according to these two criteria. A similar analysis has to be made for the language-provided patterns which manifest through a framework-specific structure or method names for this type of patterns.

2) *Security Aspects*: Developers tend to view IT security in terms of software requirements rather than taking the perspective of an attacker. Software patterns are usually chosen by developers with a particular goal in mind. For this reason, we employ general security goals for our pattern classification and not the STRIDE model, which focuses on the attacker's perspective. In the area of IT security, the most common goals are confidentiality, integrity, and availability of data [102]. **Confidentiality** guarantees the secrecy of data, whereas **integrity** makes sure that data are not modified in an unauthorized way. The *Secure Visitor* pattern fulfills the latter aspect. Nodes can only be accessed by a *Secure Visitor* who prevents unwanted access and unauthorized modifications of nodes in hierarchically structured data. **Availability** means that data/services are accessible. The *Keep Session Data in Client* pattern [92] provides the accessibility to a website if the connection between client and server is interrupted for a short time. Sometimes, **non-repudiation** (proving an action to a neutral and trustworthy third party) and **accountability** (logging certain actions for audits) are also of interest. One pattern example of these two aspects is the *Audit Interceptor* pattern [22]. It intercepts audit requests and responses to and from the business tier in JEE applications and logs them in an appropriate way. In addition, the identification of principals (e.g., users, machines, and processes), called "authentication", is important as well as **access control**, which determines which principal may access which data. Both aspects are used in the *Secure Visitor* pattern [20] where the visitor has to verify a user's credentials and check it against the access control rules for modification. As a consequence, our classification considers confidentiality, integrity, availability, non-repudiation, accounting, authentication, and access control in the second dimension.

## B. Results

We organized all software-security patterns that we described in Section III according to the aforementioned aspects (see Appendix for details). We detected that the software-security patterns often describe integrity and confidentiality problems (83 and 62 times, respectively). Authentication and access control issues are often used, too. In total, 58 and 53 patterns per criterion can be found. Lesser

attention in the software-security patterns have the security aspects availability, accountability, and non-repudiation with 22, thirteen, and two patterns, respectively, that deal with these problems. As depicted in Figure 12, we found no structural-driven pattern that covers accountability and no generic-concept pattern that handles non-repudiation aspects. All other security criteria are matched by a software-security pattern.

We detected 58 *behavioral* and 37 *structural* characterized software-security patterns. The majority of the patterns are *generic concepts*. *Generic concept* patterns cannot be directly used for a pattern recognition approach. Most of these patterns do not provide implementation information like example code that could enable a distinction into structural or behavioral. A distinction can be made if further inspections on the real usage of these pattern in software systems have been conducted and concrete implementations can be found and assessed. With this additional investigation, it may be possible to detect such patterns in the future and give designers a better idea of how to design and implement these patterns in a software system. We also expect that for some of the generic concept patterns like *Red Team The Design*, we will find no implementations in software.

## VIII. DISCUSSION

The presented application-domain classification scheme fulfills the requirements of classifications in terms of expandability, intuitive use, and is applicable for security laymen. This approach can be expanded by repeating the proceeding steps described in Section VI-A for new patterns if new application domains for security patterns emerge. The intuitive use and the applicability for security laymen is supported by the usage of only one criterion – selection by domain – which is easy to decide for a user. We suppose that a user knows in what domain she will work with security patterns, e.g., an enterprise process designer may select *Enterprise* as her application domain.

We expect that the application-domain classification helps other researchers and practitioners with specific application goals focusing on security patterns. A possible use case for this classification is, e.g., when an enterprise process architect is looking for a best practice to administer threats and risks for his enterprise. Then she can have a look at all enterprise-related patterns listed in the publications of Table III and will find patterns like *Risk Determination* and *Threat Assessment* [20] to solve her problems.

Furthermore, the second classification can help software designers to choose the right security patterns for their software system according to specific security requirements. This classification gives also a detailed overview which patterns can be used in general with respect to software related issues. Moreover, it allows one to select a software-security pattern according to its security attributes for the software design phase or determining security features for a

Recognition Security	Structural	Behavioral	Generic Concept
Accountability	_____	Audit Interceptor Secure Logger	Password Authentication
Authentication	Single Access Point Subject Description	Secure Visitor	Password Authentication
Access Control	Check Point Subject Description	Secure Visitor	Keep Session Data In Client
Availability	Partitioned Application	Secure Preforking	Keep Session Data In Client
Confidentiality	Partitioned Application	Secure Visitor	Password Authentication
Integrity	Check Point	Secure Preforking Secure Visitor	Password Authentication
Non-Repudiation	Subject Description	Audit Interceptor Secure Logger	_____

Figure 12. Our software security-pattern classification with a few examples.

security assessment.

Besides the open issue – which concrete similarities and differences design and security pattern have [11] – we identified two additional gaps in research. One is that additional work must be done to define a uniform description for security patterns to increase the description quality. As mentioned in Section V, some work has been done in this area but as we showed, the heterogeneity even by newer pattern publications is still very high. On account of this, it is desirable to have all security patterns available from a single source and presented in a uniform format like other existing open databases for design patterns (e.g., [103], [104]).

Another open issue is that we found no structural patterns with accountability aspects and no general concept patterns with non-repudiation properties. The absence of these aspects indicates a gap in the software-security pattern landscape.

Additional investigations are necessary for all software-security patterns not only *generic concept* patterns. For our classification, we were able to decide whether a pattern falls into the category of structural or behavioral patterns, but the pattern descriptions are often not sufficient and exact enough to use existing pattern recognitions out of the box for their detection. It remains a high variability in their possible implementations.

## IX. CONCLUSION AND OUTLOOK

In this paper, we presented our systematic literature review on security patterns, a comparison of design and security-pattern classifications, discussed challenges in classifying security patterns, and introduced two new classification schemes.

The first classification scheme embraces 364 published security patterns and exceeds in numbers existing classifications by far. The second classification unites the focus

of pattern recognition and security aspects. It classifies 161 software-security patterns that we obtained in the first organization process.

This classification will support our future research by the determined pattern characteristics and the indicated open issues (see Section VIII). In particular, we plan to detect and validate software-security patterns implemented in code. Automatically detected security patterns can support security and risk assessments and help in reengineering existing software systems.

## X. ACKNOWLEDGMENTS

This work was supported by the German Federal Ministry of Education and Research (BMBF) under the grant 01IS10015B (ASKS project).

## REFERENCES

- [1] M. Bunke, R. Koschke, and K. Sohr, "Application-domain classification for security patterns," in *Proceedings of the International Conferences on Pervasive Patterns and Applications*, IARIA Conferences. XPS (Xpert Publishing Services), 2011, pp. 138–143.
- [2] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Object-Oriented Software*. Addison Wesley, 1994.
- [3] J. Yoder and J. Barcalow, "Architectural patterns for enabling application security," in *Proceedings of the Conference on Pattern Languages of Programs*, Monticello/IL, 1997, pp. 1–31, last access: 23.06.2012. [Online]. Available: <http://hillside.net/plop/plop97/Proceedings/yoder.pdf>
- [4] S. Haldikis, A. Chatzigeorgiou, and G. Stephanides, "A practical evaluation of security patterns," in *Proceedings of the International Conference on Artificial Intelligence and Digital Communications*, Aug. 2006, pp. 1–8, last access: 23.06.2012. [Online]. Available: <http://inf.ucv.ro/~aidc/proceedings/2006/5%20shaldikidis.pdf>

- [5] M. Hafiz and R. E. Johnson, "Evolution of the mta architecture: the impact of security," *Software—Practice and Experience*, Wiley, vol. 38, no. 15, pp. 1569–1599, 2008.
- [6] M. Hafiz, P. Adamczyk, and R. E. Johnson, "Organizing security patterns," *IEEE Software*, vol. 24, pp. 52–60, 2007.
- [7] K. R. van Wyk and G. McGraw, "Bridging the gap between software development and information security," *Security Privacy, IEEE*, vol. 3, no. 5, pp. 75–79, Sep. 2005.
- [8] M. Laverdiere, A. Mourad, A. Hanna, and M. Debbabi, "Security Design Patterns: Survey and Evaluation," *IEEE Canadian Conference on Electrical and Computer Engineering*, pp. 1605–1608, 2006.
- [9] T. Heyman, K. Yskout, R. Scandariato, and W. Joosen, "An analysis of the security patterns landscape," in *International Workshop on Software Engineering for Secure Systems*. Washington, DC, USA: IEEE Computer Society, 2007, p. 3.
- [10] N. Yoshioka, H. Washizaki, and K. Maruyama, "A survey on security patterns," *Progress in Informatics*, vol. 5, pp. 35–47, 2008.
- [11] M. VanHilst and E. B. Fernandez, "Reverse engineering to detect security patterns in code," in *Proceedings of the International Workshop on Software Patterns and Quality*. Information Processing Society of Japan, Dec. 2007, pp. 25–30.
- [12] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern-Oriented Software Architecture: A System of Patterns*. Chichester, UK: Wiley, 1996.
- [13] E. B. Fernandez, N. Yoshioka, and H. Washizaki, "Using security patterns to build secure systems," in *Proceedings of the International Workshop on Software Patterns and Quality*. Information Processing Society of Japan, 2007, pp. 47–48.
- [14] W. Zimmer, *Pattern languages of program design*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1995, ch. Relationships between design patterns, pp. 345–364.
- [15] D. Alur, D. Malks, and J. Crupi, *Core J2EE Patterns: Best Practices and Design Strategies*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.
- [16] S. Hasso and C. Carlson, "A theoretically-based process for organizing design patterns," in *Proceedings of the Conference on Pattern Languages of Programs*, 2005, pp. 1–22, last access: 23.06.2012. [Online]. Available: [http://hillside.net/plop/2005/proceedings/PLoP2005\\_shasso0\\_3.pdf](http://hillside.net/plop/2005/proceedings/PLoP2005_shasso0_3.pdf)
- [17] N. Shi and R. A. Olsson, "Reverse engineering of design patterns from java source code," in *Automated Software Engineering*. Los Alamitos, CA, USA: IEEE Computer Society, 2006, pp. 123–134.
- [18] D. M. Kienzle, M. C. Elder, D. Tyree, and J. Edwards-Hewitt, "Security patterns repository, version 1.0," 2003, last access: 23.06.2012. [Online]. Available: <http://www.scrypt.net/~celer/securitypatterns/repository.pdf>
- [19] S. Konrad, B. H. Cheng, L. A. Campbell, and R. Wassermann, "Using security patterns to model and analyze security requirements," in *International Workshop on Requirements for High Assurance Systems*, 2003, pp. 13–22.
- [20] M. Schumacher, E. B. Fernandez, D. Hybertson, and F. Buschmann, *Security Patterns: Integrating Security and Systems Engineering*. John Wiley & Sons, 2005.
- [21] "The zachmann framework for enterprise architecture," 2012, last access: 23.06.2012. [Online]. Available: [http://zachmaninternational.com/2/Zachman\\_Framework.asp](http://zachmaninternational.com/2/Zachman_Framework.asp)
- [22] C. Steel, R. Nagappan, and R. Lai, *Core Security Patterns: Best Practices and Strategies for J2EE(TM), Web Services, and Identity Management*. Prentice Hall International, 2005.
- [23] D. G. Rosado, C. Gutiérrez, E. Fernández-Medina, and M. Piattini, "Security patterns related to security requirements," in *Proceedings of the International Workshop on Security in Information Systems*, 2006, pp. 163–173.
- [24] Commission of European Communities, "Information technology security evaluation criteria, ver. 1.2," 1991, last access: 23.06.2012. [Online]. Available: [https://www.bsi.bund.de/cae/servlet/contentblob/471346/publicationFile/30220/itsec-en\\_pdf.pdf](https://www.bsi.bund.de/cae/servlet/contentblob/471346/publicationFile/30220/itsec-en_pdf.pdf)
- [25] F. Swiderski and W. Snyder, *Threat Modeling (Microsoft Professional)*. Microsoft Press, 2004.
- [26] M. VanHilst, E. B. Fernandez, and F. A. Braz, "A multi-dimensional classification for users of security patterns," in *Proceedings of the International Workshop on Security in Information Systems*, 2008, pp. 89–98.
- [27] E. B. Fernandez, H. Washizaki, N. Yoshioka, A. Kubo, and Y. Fukazawa, "Classifying security patterns," in *Proceedings of the Asian-Pacific Web Conference*, Apr. 2008, pp. 342–347.
- [28] H. Washizaki, E. B. Fernandez, K. Maruyama, A. Kubo, and N. Yoshioka, "Improving the classification of security patterns," *Database and Expert Systems Applications*, pp. 165–170, 2009.
- [29] E. B. Fernandez, M. M. Larrondo-petrie, N. Seliya, N. Delessy, and A. Herzberg, "A pattern language for firewalls," in *Proceedings of the Conference on Pattern Languages of Programs*, Sep. 2003, pp. 1–13, last access: 23.06.2012. [Online]. Available: <http://www.hillside.net/plop/plop2003/Papers/Fernandez-firewalls.pdf>
- [30] S. R. Nelly Delessy-Gassant, Eduardo B. Fernandez and M. M. Larrondo-Petrie, "Patterns for application firewalls," in *Proceedings of the Conference on Pattern Languages of Programs*, 2004, pp. 1–19, last access: 23.06.2012. [Online]. Available: [http://www.hillside.net/plop/2004/papers/ndelessygassant0/PLoP2004\\_ndelessygassant0\\_0.doc](http://www.hillside.net/plop/2004/papers/ndelessygassant0/PLoP2004_ndelessygassant0_0.doc)
- [31] S. Henninger and V. Corrêa, "Software pattern communities: Current practices and challenges," in *Proceedings of the Conference on Pattern Languages of Programs*, ser. PLOP '07. New York, NY, USA: ACM, 2007, pp. 14:1–14:19.

- [32] "Securitypatterns.org," 2012, last access: 23.06.2012. [Online]. Available: <http://www.securitypatterns.org/>
- [33] B. Kitchenham, "Procedures for performing systematic reviews," Keele University, Keele, UK, Technical Report TR/SE-0401, 2004.
- [34] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele University, Keele, UK, Technical Report EBSE-2007-001, 2007.
- [35] IEEE, "IEEE Digital Library," 2012, last access: 23.06.2012. [Online]. Available: <http://www.computer.org/portal/>
- [36] ACM, "ACM Digital Library," 2012, last access: 23.06.2012. [Online]. Available: <http://portal.acm.org/>
- [37] The Hillside Group, "The hillside group website," 2012, last access: 23.06.2012. [Online]. Available: <http://hillside.net>
- [38] P. H. Meland and J. Jensen, "Secure software design in practice," in *Proceedings of the International Conference on Availability, Reliability and Security*, Mar. 2008, pp. 1164–1171.
- [39] M. Schumacher and U. Roedig, "Security engineering with patterns," in *Proceedings of the Conference on Pattern Languages of Programs*, 2001, pp. 1–17, last access: 23.06.2012. [Online]. Available: [http://www.hillside.net/plop/plop2001/accepted\\_submissions/PLoP2001/mschumacher0/PLoP2001\\_mschumacher0\\_1.pdf](http://www.hillside.net/plop/plop2001/accepted_submissions/PLoP2001/mschumacher0/PLoP2001_mschumacher0_1.pdf)
- [40] A. M. Braga, C. M. F. Rubira, and R. Dahab, "Tropyc: A pattern language for cryptographic software," in *Proceedings of the Conference on Pattern Languages of Programs*, 1998, pp. 1–27, last access: 23.06.2012. [Online]. Available: [http://hillside.net/plop/plop98/final\\_submissions/P25.pdf](http://hillside.net/plop/plop98/final_submissions/P25.pdf)
- [41] E. B. Fernandez, J. C. Pelaez, and M. M. Larrondo-Petrie, "Security patterns for voice over ip networks," in *International Multi-Conference on Computing in the Global Information Technology*, ser. ICCGI '07. Washington, DC, USA: IEEE Computer Society, 2007, p. 33.
- [42] K. Yskout, T. Heyman, R. Scandariato, and W. Joosen, "A system of security patterns," K.U.Leuven, Department of Computer Science, Report CW 469, Dec. 2006, last access: 23.06.2012. [Online]. Available: <http://www.cs.kuleuven.be/publicaties/rapporten/cw/CW469.abs.html>
- [43] S. T. Halkidis, A. Chatzigeorgiou, and G. Stephanides, "A qualitative analysis of software security patterns," *Computers & Security*, vol. 25, no. 5, pp. 379–392, 2006.
- [44] B. Elsinga and A. Hofman, "Security paradigm pattern language," in *Proceedings of the European Conference on Pattern Languages of Programs*. UVK - Universitaetsverlag Konstanz, 2003, pp. 363–380.
- [45] G. Dallons, P. Massonet, J.-F. Molderez, C. Ponsard, and A. Arenas, "An analysis of the chinese wall pattern for guaranteeing confidentiality in grid-based virtual organisations," in *International Workshop on Security, Trust and Privacy in Grid Systems*. IEEE, 2007, pp. 217–222.
- [46] P. Dyson and A. Longshaw, "Patterns for managing internet-technology systems," in *Proceedings of the European Conference on Pattern Languages of Programs*. UVK - Universitaetsverlag Konstanz, 2003, pp. 459–492.
- [47] B. Elsinga and A. Hofman, "Control the actor-based access rights," in *Proceedings of the European Conference on Pattern Languages of Programs*. UVK - Universitaetsverlag Konstanz, 2002, pp. 233–244.
- [48] A. M. Ernst, "Enterprise architecture management patterns," in *Proceedings of the Conference on Pattern Languages of Programs*. New York, NY, USA: ACM, 2008, pp. 1–20.
- [49] E. B. Fernandez, J. Ballesteros, A. C. Desouza-Doucet, and M. M. Larrondo-Petrie, "Security patterns for physical access control systems," in *Working Conference on Data and Applications Security*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 259–274.
- [50] S. Romanosky, "Security design patterns part 1," Nov. 2001, last access: 23.06.2012. [Online]. Available: <http://www.cgisecurity.com/lib/securityDesignPatterns.html>
- [51] A. P. Moore, M. Hanley, and D. Mundie, "A pattern for increased monitoring for intellectual property theft by departing insiders," in *Proceedings of the Conference on Pattern Languages of Programs*, 2011, pp. 1–17, last access: 23.06.2012. [Online]. Available: <http://www.hillside.net/plop/2011/papers/D-6-Moore.pdf>
- [52] D. Riehle, W. Cunningham, J. Bergin, N. Kerth, and S. Metsker, "Password patterns," in *Proceedings of the European Conference on Pattern Languages of Programs*. UVK - Universitaetsverlag Konstanz, 2002, pp. 279–288.
- [53] S. Romanosky, A. Acquisti, J. Hong, L. F. Cranor, and B. Friedman, "Privacy patterns for online interactions," in *Proceedings of the Conference on Pattern Languages of Programs*. New York, NY, USA: ACM, 2006, pp. 12:1–12:9.
- [54] M. Schumacher, "Security patterns and security standards - with selected security patterns for anonymity and privacy," in *Proceedings of the European Conference on Pattern Languages of Programs*. UVK - Universitaetsverlag Konstanz, 2002, pp. 289–300.
- [55] A. Cuevas, P. E. Khoury, L. Gomez, and A. Laube, "Security patterns for capturing encryption-based access control to sensor data," in *Proceedings of the International Conference on Emerging Security Information, Systems and Technologies*. Los Alamitos, CA, USA: IEEE Computer Society, 2008, pp. 62–67.
- [56] S. Lehtonen and J. Pärssinen, "A pattern language for cryptographic key management," in *Proceedings of the European Conference on Pattern Languages of Programs*. UVK - Universitaetsverlag Konstanz, 2002.
- [57] S. Lehtonen and J. Pärssinen, "A pattern language for key management," in *Proceedings of the Conference on Pattern Languages of Programs*, 2001, pp. 1–13, last access: 23.06.2012. [Online]. Available: [http://www.hillside.net/plop/plop2001/accepted\\_submissions/PLoP2001/slehtonen0/PLoP2001\\_slehtonen0\\_1.pdf](http://www.hillside.net/plop/plop2001/accepted_submissions/PLoP2001/slehtonen0/PLoP2001_slehtonen0_1.pdf)

- [58] K. Hashizume and E. B. Fernandez, "Symmetric encryption and xml encryption patterns," in *Proceedings of the Conference on Pattern Languages of Programs*, ser. PLoP '09. New York, NY, USA: ACM, 2009, pp. 13:1–13:8.
- [59] B. Blakley, C. Heath, and members of The Open Group Security Forum, *Security Design Patterns*. The Open Group, Apr. 2004, last access: 23.06.2012. [Online]. Available: [www.opengroup.org/onlinepubs/9299969899/toc.pdf](http://www.opengroup.org/onlinepubs/9299969899/toc.pdf)
- [60] A. Cuevas, P. E. Khoury, L. Gomez, A. Laube, and A. Sorniotti, "A security pattern for untraceable secret handshakes," in *Proceedings of the International Conference on Emerging Security Information, Systems and Technologies*, Jun. 2009, pp. 8–14.
- [61] N. Delessy and E. B. Fernandez, "Patterns for the extensible access control markup language," in *Proceedings of the Conference on Pattern Languages of Programs*, 2005, pp. 1–20, last access: 23.06.2012. [Online]. Available: [http://www.hillside.net/plop/2005/proceedings/PLoP2005\\_ndelessyandebfernandez0\\_1.pdf](http://www.hillside.net/plop/2005/proceedings/PLoP2005_ndelessyandebfernandez0_1.pdf)
- [62] M. Schumacher, "Firewall patterns," in *Proceedings of the European Conference on Pattern Languages of Programs*. UVK - Universitaetsverlag Konstanz, 2003, pp. 417–430.
- [63] N. Delessy, E. B. Fernandez, M. M. Larrondo-Petrie, and J. Wu, "Patterns for access control in distributed systems," in *Proceedings of the Conference on Pattern Languages of Programs*. New York, NY, USA: ACM, 2007, pp. 1–11.
- [64] L. B. Jr, F. L. Brown, J. Divietri, G. D. D. Villegas, and E. B. Fernandez, "The authenticator pattern," in *Proceedings of the Conference on Pattern Languages of Programs*, 1999, pp. 1–8, last access: 23.06.2012. [Online]. Available: <http://hillside.net/plop/plop99/proceedings/Fernandez4/Authenticator3.PDF>
- [65] E. B. Fernandez and R. Warriar, "Remote authenticator / authorizer," in *Proceedings of the Conference on Pattern Languages of Programs*, 2003, pp. 1–8, last access: 23.06.2012. [Online]. Available: <http://www.hillside.net/plop/plop2003/Papers/Fernandez-remote-authenticator.pdf>
- [66] M. Hafiz, "A collection of privacy design patterns," in *Proceedings of the Conference on Pattern Languages of Programs*. New York, NY, USA: ACM, 2006, pp. 1–13.
- [67] T. Okubo and H. Tanaka, "Web security patterns for analysis and design," in *Proceedings of the Conference on Pattern Languages of Programs*. New York, NY, USA: ACM, 2008, pp. 1–13.
- [68] M. Sadicoff, M. M. Larrondo-Petrie, and E. B. Fernandez, "Privacy-aware network client pattern," in *Proceedings of the Conference on Pattern Languages of Programs*, 2005, pp. 1–6, last access: 23.06.2012. [Online]. Available: [http://www.hillside.net/plop/2005/proceedings/PLoP2005\\_msadicoff0\\_0.pdf](http://www.hillside.net/plop/2005/proceedings/PLoP2005_msadicoff0_0.pdf)
- [69] B. Schleinzer and N. Yoshioka, "A security pattern for data integrity in p2p systems," in *Proceedings of the Conference on Pattern Languages of Programs*, Oct. 2010, last access: 23.06.2012. [Online]. Available: <http://www.hillside.net/plop/2010/papers/schleinzer.pdf>
- [70] P. Sommerlad, "Reverse proxy patterns," in *Proceedings of the European Conference on Pattern Languages of Programs*. UVK - Universitaetsverlag Konstanz, Jun. 2003, pp. 431–458.
- [71] S. Romanosky, "Enterprise security patterns," 2002, last access: 23.06.2012. [Online]. Available: <http://www.romanosky.net/papers/EnterpriseSecurityPatterns.pdf>
- [72] A. Kumar and E. Fernandez, "A security pattern for a virtual private network," in *Proceedings of the Latin American Conference on Pattern Languages of Programming*, 2010.
- [73] I. A. Buckley, E. B. Fernandez, and M. M. Larrondo-Petrie, "Patterns combining reliability and security," in *Proceedings of the International Conferences on Pervasive Patterns and Applications*, IARIA Conferences. XPS (Xpert Publishing Services), 2011, pp. 144–150.
- [74] C. Dougherty, K. Sayre, R. C. Seacord, D. Svoboda, and K. Togashi, "Secure design patterns," Carnegie Mellon University, Software Engineering Institute, TECHNICAL REPORT CMU/SEI 2009-TR-010, Oct. 2009, last access: 23.06.2012. [Online]. Available: [www.cert.org/archive/pdf/09tr010.pdf](http://www.cert.org/archive/pdf/09tr010.pdf)
- [75] E. B. Fernandez and J. Sinibaldi, "More patterns for operating systems access control," in *Proceedings of the European Conference on Pattern Languages of Programs*. UVK - Universitaetsverlag Konstanz, Jun. 2003, pp. 381–398.
- [76] E. B. Fernandez and T. Sorgente, "A pattern language for security models," in *Proceedings of the Conference on Pattern Languages of Programs*, 2001, pp. 1–13, last access: 23.06.2012. [Online]. Available: [http://www.hillside.net/plop/plop2001/accepted\\_submissions/PLoP2001/ebfernandezandrpan0/PLoP2001\\_ebfernandezandrpan0\\_1.pdf](http://www.hillside.net/plop/plop2001/accepted_submissions/PLoP2001/ebfernandezandrpan0/PLoP2001_ebfernandezandrpan0_1.pdf)
- [77] E. B. Fernandez, "Patterns for operating systems access control," in *Proceedings of the Conference on Pattern Languages of Programs*, 2002, pp. 1–18, last access: 23.06.2012. [Online]. Available: <http://www.hillside.net/plop/plop2002/final/OSSecPat7.doc>
- [78] E. B. Fernandez, T. Sorgente, and M. M. Larrondo-Petrie, "Even more patterns for secure operating systems," in *Proceedings of the Conference on Pattern Languages of Programs*. New York, NY, USA: ACM, 2006, pp. 1–9.
- [79] E. B. Fernandez and D. laRed Martinez, "Patterns for the secure and reliable execution of processes," in *Proceedings of the Conference on Pattern Languages of Programs*. New York, NY, USA: ACM, 2008, pp. 1–16.
- [80] E. B. Fernandez and G. Pernul, "Patterns for session-based access control," in *Proceedings of the Conference on Pattern Languages of Programs*. New York, NY, USA: ACM, 2006, pp. 8:1–8:10.
- [81] V. Gondi, "Multiple secure observers using j2ee," in *Proceedings of the Conference on Pattern Languages of Programs*, 2010, pp. 1–13, last access: 23.06.2012. [Online]. Available: <http://www.hillside.net/plop/2010/papers/gondi.pdf>

- [82] M. Hafiz, "Secure pre-forking - a pattern for performance and security," in *Proceedings of the Conference on Pattern Languages of Programs*, 2005, pp. 1–9, last access: 23.06.2012. [Online]. Available: [http://www.hillside.net/plop/2005/proceedings/PLoP2005\\_mhafiz0\\_2.pdf](http://www.hillside.net/plop/2005/proceedings/PLoP2005_mhafiz0_2.pdf)
- [83] M. Hafiz, R. E. Johnson, and R. Af, "The security architecture of gmail," in *Proceedings of the Conference on Pattern Languages of Programs*, 2004, pp. 1–9, last access: 23.06.2012. [Online]. Available: [http://www.hillside.net/plop/2004/papers/mhafiz1/PLoP2004\\_mhafiz1\\_0.pdf](http://www.hillside.net/plop/2004/papers/mhafiz1/PLoP2004_mhafiz1_0.pdf)
- [84] D. M. Kienzle and M. C. Elder, "Final technical report: Security pattern for web application development," Tech. Rep., 2002, last access: 23.06.2012. [Online]. Available: <http://www.scrip.net/~celer/securitypatterns/final%20report.pdf>
- [85] S. R. Kodituwakku, P. Bertok, and L. Zhao, "Aplrac: A pattern language for designing and implementing role-based access control," in *Proceedings of the European Conference on Pattern Languages of Programs*. UVK - Universitaetsverlag Konstanz, 2001, pp. 331–346.
- [86] Q. H. Mahmoud, "Security policy: A design pattern for mobile java code," in *Proceedings of the Conference on Pattern Languages of Programs*, 2000, pp. 1–8, last access: 23.06.2012. [Online]. Available: <http://hillside.net/plop/plop2k/proceedings/Mahmoud/Mahmoud.pdf>
- [87] H. Mouratidis, P. Giorgini, and M. Schumacher, "Security patterns for agent systems," in *Proceedings of the European Conference on Pattern Languages of Programs*. UVK - Universitaetsverlag Konstanz, Jun. 2003, pp. 399–416.
- [88] P. Morrison and E. B. Fernandez, "The credentials pattern," in *Proceedings of the Conference on Pattern Languages of Programs*. New York, NY, USA: ACM, 2006, pp. 1–4.
- [89] P. Morrison and E. B. Fernandez, "Securing the broker pattern," in *Proceedings of the European Conference on Pattern Languages of Programs*. UVK - Universitaetsverlag Konstanz, 2006, pp. 513–530.
- [90] J. L. Ortega-Arjona and E. B. Fernandez, "The secure blackboard pattern," in *Proceedings of the Conference on Pattern Languages of Programs*. New York, NY, USA: ACM, 2008, pp. 1–5.
- [91] T. Saridakis, "Design patterns for fault containment," in *Proceedings of the European Conference on Pattern Languages of Programs*. UVK - Universitaetsverlag Konstanz, 2003, pp. 493–520.
- [92] K. E. Sørensen, "Session patterns," in *Proceedings of the European Conference on Pattern Languages of Programs*. UVK - Universitaetsverlag Konstanz, 2002, pp. 301–322.
- [93] M. Weiss, "Credential delegation: Towards grid security patterns," in *Proceedings of the Nordic Conference on Pattern Languages of Programs*, 2006, pp. 65–70, last access: 23.06.2012. [Online]. Available: [http://hillside.net/vikingplop/vikingplop2006/VikingPLoP2006\\_Proceedings.pdf](http://hillside.net/vikingplop/vikingplop2006/VikingPLoP2006_Proceedings.pdf)
- [94] Y. Zhou, Q. Zhao, and M. Perry, "Policy enforcement pattern," in *Proceedings of the Conference on Pattern Languages of Programs*, 2002, pp. 1–14, last access: 23.06.2012. [Online]. Available: [http://www.hillside.net/plop/plop2002/final/ZZPerry\\_PLOP.pdf](http://www.hillside.net/plop/plop2002/final/ZZPerry_PLOP.pdf)
- [95] E. B. Fernandez, S. Mujica, and F. Valenzuela, "Two security patterns: Least privilege and secure logger/auditor," in *Proceedings of the Asian Conference on Pattern Languages of Programs*, 2011, pp. 1–12, last access: 23.06.2012. [Online]. Available: [http://patterns-wg.fuka.info.waseda.ac.jp/asianplop/proceedings2011/asianplop2011\\_submission\\_7.pdf](http://patterns-wg.fuka.info.waseda.ac.jp/asianplop/proceedings2011/asianplop2011_submission_7.pdf)
- [96] O. Ajaj and E. B. Fernandez, "A pattern for the ws-trust standard for web services," in *Proceedings of the Asian Conference on Pattern Languages of Programs*, 2010, pp. 1–11, last access: 23.06.2012. [Online]. Available: [http://patterns-wg.fuka.info.waseda.ac.jp/asianplop/proceedings2010/11-WS-Trust\\_march02-10.pdf](http://patterns-wg.fuka.info.waseda.ac.jp/asianplop/proceedings2010/11-WS-Trust_march02-10.pdf)
- [97] The Internet Society, "Point-to-point tunneling protocol (pptp)," 2012, last access: 23.06.2012. [Online]. Available: <http://tools.ietf.org/html/rfc2637>
- [98] S. T. Halkidis, N. Tsantalis, A. Chatzigeorgiou, and G. Stephanides, "Architectural risk analysis of software systems based on security patterns," *IEEE Transactions on Dependable and Secure Computing*, vol. 5, no. 3, pp. 129–142, 2008.
- [99] C. Kramer and L. Prechelt, "Design recovery by automated search for structural design patterns in object-oriented software," in *Working Conference on Reverse Engineering*. Washington, DC, USA: IEEE Computer Society, 1996, p. 208.
- [100] R. K. Keller, R. Schauer, S. Robitaille, and P. Pagé, "Pattern-based reverse-engineering of design components," in *International Conference on Software Engineering*. New York, NY, USA: ACM, 1999, pp. 226–235.
- [101] L. Wendehals, "Improving design pattern instance recognition by dynamic analysis," in *International Conference on Software Engineering*, May 2003.
- [102] R. J. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 2001.
- [103] Yahoo! Inc., "Yahoo! Design Pattern Library," last access: 23.06.2012. [Online]. Available: <http://developer.yahoo.com/yypatterns/>
- [104] Microsoft, "Microsoft patterns & practices," last access: 23.06.2012. [Online]. Available: <http://msdn.microsoft.com/en-us/practices/default>

## APPENDIX

The tables IV, V, VI and VII depict our classification of all software-security patterns that we collected during the literature-review process. The first dimension "Matching Aspects" is highlighted in light grey and a magnifier icon



Pattern Name	Matching Aspects			Security Aspects						
	Structural	Behavioral	Generic Concept	Authentication	Access Control	Integrity	Confidentiality	Non-Repudiation	Availability	Accountability
Access Control List (ACL) [63]										
Administrator Hierarchy [78]										
Capability [63]										
Check Point [3], [20]										
Checkpointed System [59], [83], [42]										
Controlled Process Creator [75], [20]										
Container Managed Security [22], [42]										
Credential [88]										
Encrypted Storage [18]										
Execution Domain [77], [20]										
Full Access With Errors [20]										
Input Validation [74]										
Multilevel Security pattern [76], [20]										
Obfuscated Transfer Object [22], [42]										
Pathname Canonicalization [74]										
Partitioned Application [18]										
Policy [59]										
Protected System [59]										
Roles [3]										
Role-Based Access Control (RBAC) [76], [20]										
Role Hierarchies [85]										
Sandbox [87]										
Secure Pipe [22], [42]										
Secure Communication [59], [42]										
Security Context [59]										
Secure Directory [74]										
Secure Process / Thread [78]										
Secure Service Facade [22], [42]										
Secure Session Object [22], [42]										
Secure Service Proxy [22]										
Session-Based Attribute-Based Authorization [80]										
Session-Based Role-Based Access Control [80]										
Single Session [85]										
Single Access Point [3], [20]										
Subject Description [59]										
Symmetric Encryption [58]										
XML Encryption Pattern [58]										
A Pattern for WS-Trust [96]										
Access Controller [87]										
Account Lockout [18]										
Agent Authenticator [87]										
Agency Guard [87]										
Audit Interceptor [22]										

Table IV  
SOFTWARE-SECURITY PATTERNS CLASSIFIED BY SECURITY ASPECTS AND RECOGNITION NEEDS (1).

Pattern Name	Matching Aspects			Security Aspects						
	Structural	Behavioral	Generic Concept	Authentication	Access Control	Integrity	Confidentiality	Non-Repudiation	Availability	Accountability
Authenticator [64], [59], [20]										
Authentication Enforcer [22]										
Authorization Pattern [76]										
Authorization Enforcer [22]										
Assertion Builder Pattern [22]										
Controlled Object Factory [20]										
Controlled Object Monitor [75] [20]										
Controlled Virtual Address Space [75]										
Credential Delegation [93]										
Credential Tokenizer [22]										
Defer to Kernel [74]										
Dynamic Service Management [22]										
File Authorization [77], [20]										
Full View With Errors [3]										
Grant-Based Access Control Pattern (GBAC) [55]										
ID/Password Authentication [67]										
Information Obscurity [20]										
Intercepting Validator [22]										
Intercepting Web Agent [22]										
Known Partners [20]										
Limited Access [20]										
Limited View [3]										
Message Inspector [22]										
Message Interceptor Gateway [22]										
Multiple Secure Observers Using J2EE [81]										
Network Address Blacklist [18]										
Password Synchronizer Pattern [22]										
Policy-Based Access Control [63]										
Policy Delegate [22]										
Policy Enforcement Pattern [94]										
Privilege Separation (PrivSep) [74]										
Protected Entry Points [79]										
Protection Rings [79]										
Secure Base Action [22]										
Secure Logger [22]										
Secure Message Router [22]										
Single Sign-on Delegator Pattern [22]										
Session [3]										
Security Policy: A Design Pattern For Mobile Java Code [86]										
Secure Broker Pattern [89]										
Security Session [20]										
Session Timeout [92], [42]										



Table V  
SOFTWARE-SECURITY PATTERNS CLASSIFIED BY SECURITY ASPECTS AND RECOGNITION NEEDS (2).

Pattern Name	Matching Aspects			Security Aspects						
	Structural	Behavioral	Generic Concept	Authentication	Access Control	Integrity	Confidentiality	Non-Repudiation	Availability	Accountability
Sealed And Signed Envelope [57]										
Sealed Envelope [57]										
Security Association [59]										
Secure Builder Factory [74]										
Secure Chain of Responsibility [74]										
Secure Factory [74]										
Secure State Machine [74]										
Secure Strategy Factory [74]										
Secure Visitor [74]										
Secure Preforking [82]										
Virtual Address Space Access Control [77]										
Access Session [80]										
Access Control requirements [20]										
Actor and Role Lifecycle [47]										
Address Book [57]										
Administrator Objects [85]										
Alice And Friends [57]										
Authenticated Session [18]										
Authorization [20]										
Build The Server From The Ground Up [18]										
Clear Sensitive Information [74]										
Client Data Storage [18]										
Client Input Filters [18]										
Choose The Right Stuff [18]										
Compartmentalization [83]										
Content Independent Processing [83]										
Controlled Execution Environment [77]										
Demilitarized Zone [20]										
Directed Session [18]										
Distributed Responsibility [83]										
Distrustful Decomposition [74]										
Document The Security Goals [18]										
Document The Server Configuration [18]										
Enroll By Validating Out Of Band [18]										
Enroll Using Third-Party Validation [18]										
Enroll With A Pre-Existing Shared Secret [18]										
Enroll Without Validating [18]										
Face-To-Face [57]										
Fault Container [91]										
Front Door [20]										
Hidden Implementation [18]										
Input Guard [91]										

Table VI  
SOFTWARE-SECURITY PATTERNS CLASSIFIED BY SECURITY ASPECTS AND RECOGNITION NEEDS (3).

Pattern Name	Matching Aspects			Security Aspects						
	Structural	Behavioral	Generic Concept	Authentication	Access Control	Integrity	Confidentiality	Non-Repudiation	Availability	Accountability
Keep Session Data In Client [92]										
Keep Session Data In Server [92]										
Key In The Pocket [57]										
Load Balancer [92], [42]										
Log For Audit [18]										
Minefield [18]										
Multilevel Secure Partitions [79]										
Output Guard [91]										
Password Authentication [18]										
Password Propagation [18]										
Patch Proactively [18]										
Privilege-Limited Role [85]										
Reference Monitor [77]										
Red Team The Design [18]										
Resource Acquisition Is Initialization (RAII) [74]										
Role Based Access [85]										
Role Validator [85]										
Secure Access Layer [3]										
Secure Assertion [18]										
Secure Channels [20]										
Server Sandbox [18]										
Session Failover [92], [42]										
Session Management [67]										
Session Scope [92]										
Seal Ring Engraver [57]										
Signed Envelope [57]										
Share Responsibility For Security [18]										
Subject Descriptor [20]										
Test On A Staging Server [18]										
The Forged Seal Ring [57]										
The Real Thing [57]										
There Is Somebody Eavesdropping [57]										
Trusted Proxy [18]										
Unique Entry of Information [83]										
Validated Transactions [18]										
Virtual Address Space Structure Selection [78]										

Table VII  
SOFTWARE-SECURITY PATTERNS CLASSIFIED BY SECURITY ASPECTS AND RECOGNITION NEEDS (4).

 indicates that a pattern belongs to the aspect of this dimension. The second dimension "Security Aspects" is highlighted with a grey background and a lock  shows which security aspects a pattern addresses. Some patterns were described by more than one publication. Therefore, we put all publications that describe the pattern with this name in the order of their publication year behind the pattern name.



[www.iariajournals.org](http://www.iariajournals.org)

**International Journal On Advances in Intelligent Systems**

✦ ICAS, ACHI, ICCGI, UBICOMM, ADVCOMP, CENTRIC, GEOProcessing, SEMAPRO, BIOSYSCOM, BIOINFO, BIOTECHNO, FUTURE COMPUTING, SERVICE COMPUTATION, COGNITIVE, ADAPTIVE, CONTENT, PATTERNS, CLOUD COMPUTING, COMPUTATION TOOLS, ENERGY, COLLA, IMMM, INTELLI, SMART, DATA ANALYTICS

✦ issn: 1942-2679

**International Journal On Advances in Internet Technology**

✦ ICDS, ICIW, CTRQ, UBICOMM, ICSNC, AFIN, INTERNET, AP2PS, EMERGING, MOBILITY, WEB

✦ issn: 1942-2652

**International Journal On Advances in Life Sciences**

✦ eTELEMED, eKNOW, eL&mL, BIODIV, BIOENVIRONMENT, BIOGREEN, BIOSYSCOM, BIOINFO, BIOTECHNO, SOTICS, GLOBAL HEALTH

✦ issn: 1942-2660

**International Journal On Advances in Networks and Services**

✦ ICN, ICNS, ICIW, ICWMC, SENSORCOMM, MESH, CENTRIC, MMEDIA, SERVICE COMPUTATION, VEHICULAR, INNOV

✦ issn: 1942-2644

**International Journal On Advances in Security**

✦ ICQNM, SECURWARE, MESH, DEPEND, INTERNET, CYBERLAWS

✦ issn: 1942-2636

**International Journal On Advances in Software**

✦ ICSEA, ICCGI, ADVCOMP, GEOProcessing, DBKDA, INTENSIVE, VALID, SIMUL, FUTURE COMPUTING, SERVICE COMPUTATION, COGNITIVE, ADAPTIVE, CONTENT, PATTERNS, CLOUD COMPUTING, COMPUTATION TOOLS, IMMM, MOBILITY, VEHICULAR, DATA ANALYTICS

✦ issn: 1942-2628

**International Journal On Advances in Systems and Measurements**

✦ ICQNM, ICONS, ICIMP, SENSORCOMM, CENICS, VALID, SIMUL, INFOCOMP

✦ issn: 1942-261x

**International Journal On Advances in Telecommunications**

✦ AICT, ICDT, ICWMC, ICSNC, CTRQ, SPACOMM, MMEDIA, COCORA, PESARO, INNOV

✦ issn: 1942-2601